

Bio-inspired Algorithms for Autonomous Deployment and Localization of Sensor Nodes

Raghavendra V. Kulkarni, *Senior Member, IEEE*, and Ganesh Kumar Venayagamoorthy, *Senior Member, IEEE*

Abstract—Optimal deployment and accurate localization of sensor nodes have a strong influence on the performance of a wireless sensor network (WSN). This paper considers real-time autonomous deployment of sensor nodes from an unmanned aerial vehicle (UAV). Such a deployment has importance, particularly in *ad hoc* WSNs, for emergency applications, such as disaster monitoring and battlefield surveillance. The objective is to deploy the nodes only in the terrains of interest, which are identified by segmentation of the images captured by a camera on board the UAV. Bioinspired algorithms, particle swarm optimization (PSO) and bacterial foraging algorithm (BFA), are presented in this paper for image segmentation. In addition, PSO and BFA are presented for distributed localization of the deployed nodes. Image segmentation for autonomous deployment and distributed localization are formulated as multidimensional optimization problems, and PSO and BFA are used as optimization tools. Comparisons of the results of PSO and BFA for autonomous deployment and distributed localization are presented. Simulation results show that both the algorithms perform multilevel image segmentation faster than the exhaustive search for optimal thresholds. Besides, PSO-based localization is observed to be faster, and BFA-based localization is more accurate.

Index Terms—Bacterial foraging algorithm (BFA), image thresholding, node localization, particle swarm optimization (PSO), wireless sensor networks (WSNs).

I. INTRODUCTION

WIRELESS sensor networks (WSNs) are networks of distributed autonomous nodes that can sense or monitor physical or environmental conditions cooperatively [1]. WSNs are used in many applications, such as surveillance, environment and habitat monitoring, structural health monitoring, healthcare, home automation, and traffic control. It is a common practice in surveillance and monitoring applications that WSN nodes are dropped from airborne vehicles, such as helicopters. However, this strategy cannot be used when the area to be monitored is a hostile or dangerous territory. Autonomous deployment of sensor nodes using an unmanned aerial vehicle (UAV) is proposed to circumvent this limitation [2]. Platforms for autonomous deployment using UAVs are in existence [3]. Modern UAVs are

Manuscript received May 10, 2009; revised August 30, 2009 and January 13, 2010; accepted April 21, 2010. This work was supported by the National Science Foundation, USA under Grant SENSORS: ECCS 0529292. This paper was recommended by Associate Editor S. H. Rubin.

The authors are with the Real-Time Power and Intelligent Systems Laboratory, Missouri University of Science and Technology, Rolla, MO 65409 USA (e-mail: arvie@ieee.org; gkumar@ieee.org).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCC.2010.2049649

endowed with control and perception that make them capable of coordinated deployment missions [4].

Computer vision is the most relevant perception to UAVs. It is used for motion and position estimation, object detection and tracking, autonomous takeoff and landing, as well as for applications, such as detection, monitoring, and terrain mapping. Terrain recognition by means of image segmentation has relevance to autonomous deployment of WSN nodes. This can be effectively used to prevent sensor nodes from being deployed into the areas of no interest (water or fire in the terrain, for example). Such a prevention is necessary from the point of view of conservation of the environment. Besides, it reduces the loss of sensor nodes and ensures that the required sensing coverage can be achieved with less number of nodes deployed only in the terrains of interest. It also helps to eliminate congestion and delay caused by redundant traffic from the nodes in the areas of no interest.

Most machine vision methods use image segmentation as an important preprocessing technique. Thresholding method is commonly used for segmentation of an image into two or more classes. Let $f(x, y)$ denote a grayscale image of size $H \times W$ pixels that has L intensity levels. Two-level thresholding deals with determining a value of threshold t to perform the operation expressed by (1) for $x = 1, 2, \dots, H$ and $y = 1, 2, \dots, W$

$$F(x, y) = \begin{cases} 0, & \text{if } f(x, y) \leq t \\ L, & \text{if } f(x, y) > t. \end{cases} \quad (1)$$

This can be extended to three-level thresholding, in which there exist two threshold levels t_1 and t_2 such that $t_1 < t_2$, and the thresholding operation is performed, as expressed in the following equation:

$$F(x, y) = \begin{cases} 0, & \text{if } f(x, y) \leq t_1 \\ \frac{1}{2}(t_1 + t_2), & \text{if } t_1 < f(x, y) \leq t_2 \\ L, & \text{if } f(x, y) > t_2. \end{cases} \quad (2)$$

This can be further extended to generic n -level thresholding in which $n - 1$ threshold levels t_1, t_2, \dots, t_{n-1} are necessary. It is obvious from (1) and (2) that the effectiveness of multi-level segmentation largely depends on the values of threshold levels t_1, t_2, \dots, t_{n-1} . The basic question is: how are the threshold levels determined? Many methods have been proposed in literature, a survey of which is presented in [5].

Otsu criterion is popular in automatic thresholding [6]. This method uses statistical variance of distribution of pixels in each class as the objective function. A thresholding method that uses this criterion evaluates a number of candidate threshold levels for the chosen objective function, and chooses the one that has

the best objective function value. An obvious question is: how to choose the candidate threshold levels? An intuitive solution is to try each possible combination of threshold levels. Though straightforward, this involves a large number of objective function evaluations, which renders the method unsuitable in many applications. Clearly, it is an optimization problem, which has to be solved in real time.

Location is critically important in WSNs for monitoring and tracking applications. Location information of the nodes is used to detect and record events or to route packets using geometric-aware routing [7], [8]. Sometimes, location itself is the data that needs to be sensed. Equipping each node with a GPS is not an attractive solution because of cost, size, and energy constraints. Node localization, which refers to determining locations of all deployed sensors, is an area of active research. Many localization algorithms available in literature share a common feature that they estimate the locations of the nodes using *a priori* knowledge of the coordinates of special nodes called beacons, landmarks, or anchors [9].

A WSN consists of N nodes, each having a communication range of r , distributed in a 2-D mission field. The WSN is represented as the Euclidean graph $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of sensor nodes. $\langle i, j \rangle \in E$ if the distance between v_i and v_j is $d_{ij} \leq r$. Unknown nodes (also known as free or dumb nodes) are the set \mathcal{U} of nonbeacon nodes that do not know their localization information. Settled nodes are the set \mathcal{S} of nodes that managed to estimate their positions using the localization algorithm. Given a WSN $G = (V, E)$, and a set of beacon nodes \mathcal{B} and their positions (x_b, y_b) , for all $b \in \mathcal{B}$, it is desired to find the position (x_u, y_u) of as many $u \in \mathcal{U}$ as possible, transforming the unknown nodes into settled nodes \mathcal{S} .

WSN localization is a two-phase process. In the first phase, known as ranging phase, nodes estimate their distances from beacons (or settled nodes) using the signal propagation time or the strength of the received signal. The signal propagation time is estimated through the measurement of the time of arrival, the round trip time of flight, or the time difference of arrival of the signal. Precise measurement of these parameters is not possible due to noise; therefore, results of the location algorithms that depend on these parameters are likely to be inaccurate. In the second phase, position estimation of the target nodes is carried out using the ranging information. This is done either by solving a set of simultaneous equations, or by using an optimization algorithm that minimizes the localization error. In iterative localization algorithms, the settled nodes serve as beacons and the localization process is repeated until either all nodes are settled, or no more nodes can be localized.

The WSN localization problem has been tackled by several interesting approaches in literature. A survey of localization systems for ubiquitous computing is presented in [10]. An overview of localization systems for WSNs is presented in [9]. An overview of the measurement techniques in sensor network localization and the one-hop localization algorithms based on the measurements is presented in [11]. An efficient localization system that extends GPS capabilities to non-GPS nodes in an *ad hoc* network is proposed in [12], in which beacons flood their location information to all nodes in the network. Each dumb node

estimates its location by means of its estimated distance from three or more beacons. A refinement to this approach is proposed in [13], in which the nodes improve their localization accuracy by measuring their distances from their neighbors. The issue of error accumulation is addressed in [14] through Kalman-filter-based least-square estimation. The node localization problem is addressed using convex optimization based on semi-definite programming in [15]. The semi-definite programming approach is further extended to nonconvex inequality constraints in [16] and to a gradient-search technique in [17].

The primary contributions of this paper are as follows.

- 1) Bioinspired algorithms, particle swarm optimization (PSO) [18] and bacterial foraging algorithm (BFA) [19], are presented for thresholding of the terrain images captured from a downward-pointing camera on board the UAV used for autonomous deployment of WSN nodes.
- 2) A comparative performance analysis of the algorithms is presented. Both algorithms perform multilevel image segmentation faster than the exhaustive search method.
- 3) The same algorithms are proposed for postdeployment-distributed node localization in the WSN.
- 4) A comparative analysis of the algorithms is presented in terms of accuracy of localization and computing time. The results show that the BFA-based localization is more accurate, and the PSO-based localization is faster.

The rest of this paper is organized as follows. Brief reviews of PSO and BFA are presented in Section II-A and II-B, respectively. PSO- and BFA-based image thresholding methods for autonomous deployment of sensor nodes from a UAV are presented in Section III. PSO- and BFA-based iterative localization is discussed in Section IV. Details of the MATLAB-based numerical simulations are presented and the results are discussed in Section V. Finally, concluding remarks are given in Section VI.

II. BIOINSPIRED ALGORITHMS: A BRIEF REVIEW

Real-world problems in many disciplines of engineering are formulated as multidimensional optimization problems. Analytical methods to solve optimization problems require enormous computational efforts, which grow exponentially as the problem size increases. This is the motivation for optimization methods that require moderate memory and computational resources, and yet produce good results. Researchers have used bioinspired stochastic optimization methods as computationally efficient alternatives to analytical methods [20]. Examples of bioinspired optimization algorithms include PSO, BFA, genetic algorithm (GA) [21], and differential evolution (DE) [22]. Hybrids of these algorithms have been developed in order to achieve fast convergence or better solution quality. Examples of hybrid algorithms include those between PSO and GA [23], PSO and DE [24], BFA and GA [25], and BFA and PSO [26], [27]. However, these advanced algorithms involve additional computational overheads, which may be prohibitive in resource-constrained computational units, such as wireless sensor nodes. Therefore, basic versions of PSO and BFA are applied for thresholding-based autonomous deployment and distributed node localization in this study. Ease of implementation and quick convergence are the advantages

of PSO, while high quality of solution and moderate memory requirement are the advantages of BFA. The following sections present an overview of PSO and BFA.

A. Particle Swarm Optimization

PSO is a population-based iterative search algorithm that models social behavior of a flock of birds [28], [29]. It has been found effective in solving several kinds of multidimensional optimization problems. Since its introduction in [18], PSO has seen many modifications and has been adapted to different complex environments [30]. Many versions of PSO have been proposed and applied in areas, including multirobot navigation [31], power systems [20], pattern classification [32], and electromagnetics [33].

PSO consists of a population (or a swarm) of s particles, each of which is a candidate solution. The particles explore an n -dimensional hyperspace in search of the global solution, where n represents the number of parameters to be optimized. The d th dimension of the particle i has the position X_{id} and the velocity V_{id} , $1 \leq i \leq s$ and $1 \leq d \leq n$. The particles are initially assigned random positions and velocities within fixed boundaries, i.e., $X_{\min} \leq X_{id} \leq X_{\max}$ and $-V_{\max} \leq V_{id} \leq V_{\max}$. Each particle in the swarm is evaluated by an objective function $f(x_1, x_2, \dots, x_n)$, where $f: \mathbb{R}^n \rightarrow \mathbb{R}$. The cost (or the fitness) of a particle is determined from its position in the search space. The cost of a particle closer to the global solution is lower than that of a particle that is farther. Alternately, the fitness of a particle closer to the global solution is higher than that of a particle that is farther. PSO thrives to minimize a cost function, or maximize a fitness function. In each iteration, velocities and positions of all particles are updated to persuade them to achieve a lower cost (or a higher fitness).

In the global-best version of PSO, each particle has a memory to store $Xpbest_{id}$, the position where it had the lowest cost, and $Xgbest_d$, the position of the best particle in the population [18]. At each iteration k , velocity v_{id} and position X_{id} of each particle are updated using (3) and (4)

$$V_{id}(k) = wV_{id}(k-1) + c_1r_{1id}(k)(Xpbest_{id} - X_{id}) + c_2r_{2id}(k)(Xgbest_d - X_{id}) \quad (3)$$

$$X_{id}(k) = X_{id}(k-1) + V_{id}(k). \quad (4)$$

Here, r_1 and r_2 are the random numbers with a uniform distribution in the range $[0, 1]$. The velocity update equation (3) shows that a particle is influenced by three components of acceleration. The first term has the inertia weight w , $0.2 < w < 1.2$, which denotes the inertia of the particle. Better PSO performance has been observed with dynamically changing V_{\max} [34] and by linearly decreasing the value of w in each iteration [35]. The pseudocode for the global-best version of PSO for maximization of a fitness function is given in Algorithm 1.

B. Bacterial Foraging Algorithm

BFA is a newly introduced evolutionary optimization algorithm that mimics the foraging behavior of *Escherichia coli* (commonly referred to as *E. coli*) bacteria. BFA was first in-

Algorithm 1 The global-best version of PSO for minimization of a cost function $f(\cdot)$

```

1: Initialize  $w$ ,  $c_1$  and  $c_2$ 
2: Initialize maximum allowable iterations  $k_{\max}$ 
3: Initialize the target fitness  $f_T$ 
4: Initialize  $X_{\min}$ ,  $X_{\max}$ ,  $V_{\min}$  and  $V_{\max}$ 
5: Initialize  $Xgbest$  such that  $f(Xgbest)$  is as close to  $\infty$  as possible.
6: for each particle  $i$  do
7:   for each dimension  $d$  do
8:     Initialize  $X_{id}$  randomly:  $X_{\min} \leq X_{id} \leq X_{\max}$ 
9:      $Xpbest_{id} = X_{id}$ 
10:    Initialize  $V_{id}$  randomly:  $V_{\min} \leq V_{id} \leq V_{\max}$ 
11:  end for
12:  Compute  $f(X_i)$ 
13:  if  $f(X_i) < f(Xgbest)$  then
14:    for each dimension  $d$  do
15:       $Xgbest_d = X_{id}$ 
16:    end for
17:  end if
18: end for
19: Iteration  $k = 1$ 
20: while ( $k \leq k_{\max}$ ) AND ( $f(Xgbest) > f_T$ ) do
21:   for each particle  $i$  do
22:    for each dimension  $d$  do
23:      Compute velocity  $V_{id}(k)$  using (3)
24:      Restrict  $V_{id}$  to  $V_{\min} \leq V_{id} \leq V_{\max}$ 
25:      Compute position  $X_{id}(k)$  using (4)
26:      Restrict  $X_{id}$  to  $X_{\min} \leq X_{id} \leq X_{\max}$ 
27:    end for
28:    Compute  $f(X_i)$ 
29:    if  $f(X_i) < f(Xpbest_i)$  then
30:      for each dimension  $d$  do
31:         $Xpbest_{id} = X_{id}$ 
32:      end for
33:    end if
34:    if  $f(X_i) < f(Xgbest)$  then
35:      for each dimension  $d$  do
36:         $Xgbest_d = X_{id}$ 
37:      end for
38:    end if
39:  end for
40:   $k = k + 1$ 
41: end while

```

troduced in [19]. There are successful applications of BFA in optimization problems, such as economic load dispatch [27] and power systems [36], [37].

BFA models the movement of *E. coli* bacteria that thrive to find nutrient-rich locations in human intestine. An *E. coli* bacterium moves using a pattern of two types of movements: tumbling and swimming. Tumbling refers to a random change in the direction of movement, and swimming refers to moving in a straight line in a given direction. A bacterium in a neutral medium alternates between tumbling and swimming movements.

Algorithm 2 Bacterial Foraging Algorithm for minimization of a cost function $J(\cdot)$

```

1: Initialize  $p, S, N_c, N_{re}, N_{ed}, p_{ed}, N_s, d_a, w_a, h_r$  and  $w_r$ 
2: Initialize  $P_i$  randomly for  $i = 1, 2, \dots, S$ 
3: Initialize  $C(i)$  for  $i = 1, 2, \dots, S$ 
4: Set the loops indexes  $j, k$  and  $l$  to 0.
5: //Elimination-Dispersal loop:
6: while  $l \leq N_{ed}$  do
7:    $l = l + 1$ 
8:   //Reproduction loop:
9:   while  $k \leq N_{re}$  do
10:     $k = k + 1$ 
11:    //Chemotaxis loop:
12:    while  $j \leq N_c$  do
13:       $j = j + 1$ 
14:      for each bacterium  $i = 1, 2, \dots, S$  do
15:        Compute  $J(i, j, k, l)$ 
16:        Let  $J(i, j, k, l) = J(i, j + 1, k, l) +$ 
           $J_{cc}(P_i(j, k, l), \mathcal{P}(j, k, l))$ 
17:        Let  $J_{last} = J(i, j, k, l)$ 
18:        //Tumble:
19:        Generate a  $p$ -dimensional random vector
           $\Delta_m(i), i = 1, 2, \dots, p$  on  $[-1, 1]$ 
20:        //Move:
21:        Let  $P_i = P_i(j + 1, k, l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$ 
22:        Let  $J(i, j + 1, k, l) = J(i, j + 1, k, l) +$ 
           $J_{cc}(P_i(j + 1, k, l), \mathcal{P}(j + 1, k, l))$ 
23:        //Swim:
24:        Let  $m = 0$ 
25:        while  $m < N_s$  do
26:          Let  $m = m + 1$ 
27:          if  $J(i, j + 1, k, l) > J_{last}$  then
28:            Let  $J(i, j + 1, k, l) = J_{last}$ 
29:            Let  $P_i = P_i(j + 1, k, l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$ 
30:            Use this  $P_i$  to compute new  $J(i, j + 1, k, l)$ 
31:          else
32:             $m = N_s$ 
33:          end if
34:        end while
35:      end for
36:    end while
37:    Compute for each bacterium  $i$ , for given  $k$  and  $l$ 
38:       $J_{health}^i = \sum_{j=1}^{N_c+1} J(i, j, k, l)$ 
39:    Eliminate  $S_r$  fraction of bacteria with highest  $J_{health}$ 
    and split the other bacteria into two at their locations.
40:  end while
41:  For each bacterium, with probability  $P_d$  eliminate the
    bacterium and create a new one at a random position.
42: end while

```

Suppose it is desired to search for the position P in a p -dimensional space, where function $J(P)$, $P \in \mathbb{R}^p$ has the global minimum. Let P_i be the initial position of bacterium i in the search space, $i = 1, 2, \dots, S$, where S is the number of bacteria. In biological bacteria populations, S can be as high as 10^9 and p is three. Let $J(P_i)$ represent an objective function. Let

$J(P_i) < 0$, $J(P_i) = 0$, and $J(P_i) > 0$ represent the bacterium at location P_i in nutrient rich, neutral, and noxious environments, respectively. Chemotaxis is a foraging behavior that captures the process of optimization, where bacteria try to climb up the nutrient concentration gradient (i.e., bacteria try to achieve positions having lower values of $J(P_i)$ and avoid being at positions P_i , where $J(P_i) \geq 0$) [19].

The bacterium i at position P_i takes a chemotactic step j with the step size $C(i)$ and evaluates itself for objective function $J(P_i)$ at each step. If at position $P_i(j + 1)$, the value J is better than at position $P_i(j)$, then another step of same size $C(i)$ in the same direction will be taken again, if that step resulted in a position with a better value than at the previous step. This is referred to as a swimming step. Swimming is continued until a minimum fitness value is reached, but only for a maximum number of steps N_s . After N_c chemotactic steps, a reproduction step is taken in which the population is sorted in ascending order of the objective function value J and the least healthy bacteria are replaced by the copies of the healthier bacteria. After N_{re} reproduction steps, an elimination-dispersal step is taken. Here, a bacterium is eliminated and a new bacterium is created at a random location in the search space with probability p_{ed} . The optimization stops after N_{ed} elimination-dispersal rounds.

Bacteria create swarms by means of cell-to-cell signaling via an attractant and a repellent. Cell-to-cell attraction for bacterium i is represented with $J_{cc}(\mathcal{P}, P_i)$, $i = 1, 2, \dots, S$. This is defined in (5) as follows:

$$\begin{aligned}
J_{cc}(\mathcal{P}, P_i) = & \sum_{t=1}^S \left[-d_a \exp \left(-w_a \sum_{m=1}^p (P_{i,m} - P_{t,m})^2 \right) \right] \\
& + \sum_{t=1}^S \left[-h_r \exp \left(-w_r \sum_{m=1}^p (P_{i,m} - P_{t,m})^2 \right) \right].
\end{aligned} \tag{5}$$

Here, $J_{cc}(P_i, \mathcal{P})$ denotes the combined cell-to-cell attraction and repulsion effects for bacterium i at position $P_i = [P_{i,1}, P_{i,2}, \dots, P_{i,m}]^T$ and the whole swarm of bacteria $\mathcal{P} = \{P_1, P_2, \dots, P_S\}$. The cell-to-cell signaling $J_{cc}(\cdot)$ helps cells to move toward other cells, but not very close to them. Here, h_r and w_r are height and width of the repellent, respectively, and d_a and w_a are depth and width of the attractant, respectively. For BFA, the maximum number of objective function evaluations is $N_{ed} \cdot N_{re} \cdot N_c \cdot S \cdot N_s$. A general biologically inspired thumb-of-rule for choosing the parameters of BFA is: $N_c > N_{re} > N_{ed}$ [27]. A detailed pseudocode for BFA is given in Algorithm 2.

In this paper, PSO and BFA are applied to the problem of image thresholding for autonomous deployment and postdeployment localization of WSN nodes. The overall flowchart is shown in Fig. 1. PSO- and BFA-based image thresholding and localization are discussed in Sections III and IV.

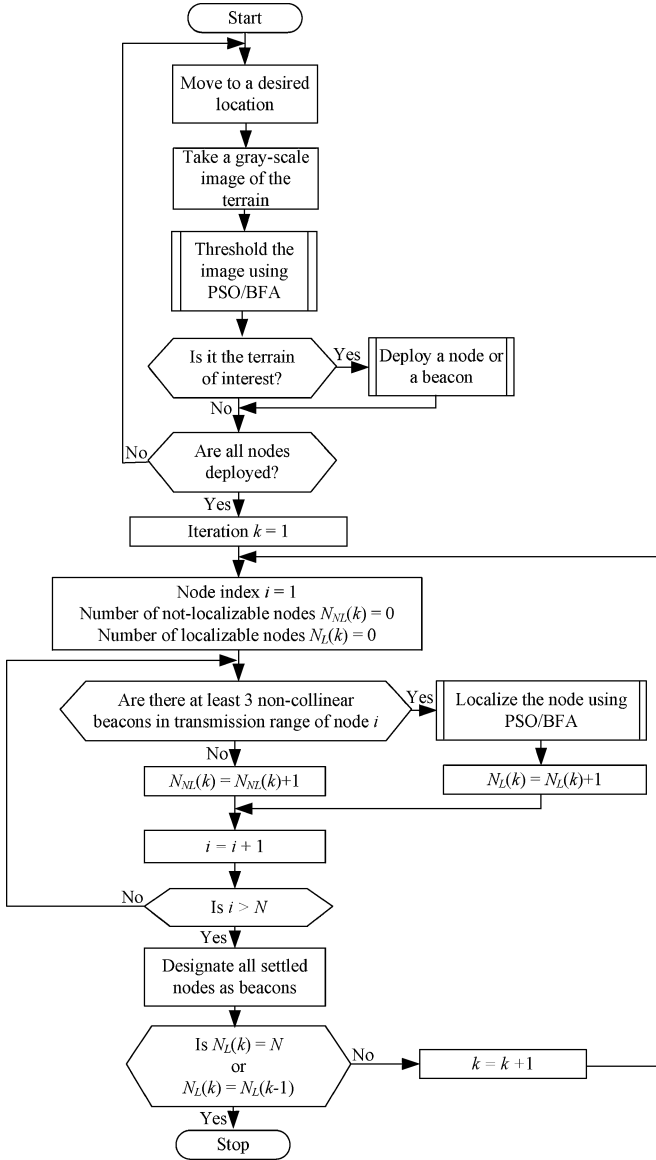


Fig. 1. Flowchart for bioinspired autonomous deployment and distributed localization of sensor nodes.

III. PSO- AND BFA-BASED IMAGE THRESHOLDING FOR AUTONOMOUS DEPLOYMENT

There exist numerous image segmentation methods that are classified into the following categories based on the image information they exploit:

- 1) histogram-shape-based methods;
- 2) clustering-based methods;
- 3) entropy-based methods;
- 4) object-attribute-based methods;
- 5) spatial methods;
- 6) local methods.

A detailed survey of 40 image thresholding algorithms and their relative quantitative performance analysis is presented in [5]. The PSO- and BFA-based thresholding methods used here exploit the image histogram shape.

Otsu proposed a nonparametric and unsupervised method of automatic threshold selection for image segmentation in [6]. This method establishes three appropriate criteria for evaluating the suitability of a given threshold level from the image histogram. The following paragraphs discuss the basics of the Otsu evaluation criteria and point out strengths and weaknesses of the Otsu-based exhaustive search. Consider a digital image having a height of H pixels and a width of W pixels, in which the intensities are represented in L gray levels $[1, 2, \dots, L]$. Let n_i represent the number of pixels having intensity level i . It can be observed that the total number of pixels N satisfies $N = H \times W = n_1 + n_2 + \dots + n_L$. The 1-D vector n_i with $i = [1, 2, \dots, L]$ represents the image histogram. The histogram is normalized and regarded as a probability distribution, as in (6), as follows:

$$p_i = \frac{n_i}{N}, \quad p_i > 0 \quad \text{and} \quad \sum_{i=1}^L p_i = 1. \quad (6)$$

Suppose it is desired to dichotomize the pixels into classes C_1 , which represents background, and C_2 , which represents an object, using a threshold level t . The class C_1 contains all pixels having intensities less than or equal to t , and the class C_2 contains all pixels having intensities greater than t . The probabilities of occurrence classes C_1 and C_2 are given by (7) and (8), respectively

$$\omega_1(t) = \Pr(C_1) = \sum_{i=1}^t p_i \quad (7)$$

$$\omega_2(t) = \Pr(C_2) = \sum_{i=t+1}^L p_i. \quad (8)$$

The mean levels of classes C_1 and C_2 are given by (9) and (10), respectively

$$\mu_1(t) = \frac{\sum_{i=1}^t ip_i}{\omega_1} \quad (9)$$

$$\mu_2(t) = \frac{\sum_{i=t+1}^L ip_i}{\omega_2}. \quad (10)$$

These probabilities and mean levels satisfy the conditions $\omega_1(t) + \omega_2(t) = 1$ and $\omega_1\mu_1 + \omega_2\mu_2 = \mu_T$, where

$$\mu_T = \mu(L) = \sum_{i=1}^L ip_i \quad (11)$$

is the total mean level of the image. The variance of distribution of pixels in classes C_1 and C_2 are given by (12) and (13), respectively

$$\sigma_1^2(t) = \sum_{i=1}^t \{i - \mu_1(t)\}^2 \frac{p_i}{\omega_1} \quad (12)$$

$$\sigma_2^2(t) = \sum_{i=t+1}^L \{i - \mu_2(t)\}^2 \frac{p_i}{\omega_2}. \quad (13)$$

In order to evaluate the goodness of the threshold at level t , Otsu introduced three objective functions (λ), (κ), and ($t\eta$), defined in (14), as follows:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \quad \kappa = \frac{\sigma_T^2}{\sigma_W^2}, \quad \text{and} \quad \eta = \frac{\sigma_B^2}{\sigma_T^2} \quad (14)$$

where σ_W^2 is the within-class variance, σ_B^2 is the between-class variance, and σ_T^2 is the total variance. These are defined in (15)–(17), respectively

$$\sigma_W^2 = \omega_1 \sigma_1^2 + \omega_2 \sigma_2^2 \quad (15)$$

$$\sigma_B^2 = \omega_1 \omega_2 + (\mu_2 - \mu_1)^2 \quad (16)$$

$$\sigma_T^2 = \sum_{i=1}^L (i - \mu_T)^2 p_i. \quad (17)$$

With this, the problem of two-level thresholding is reduced to an optimization problem to search for the threshold t^* that maximizes one of the objective functions defined in (14). It also follows that the threshold t^* that maximizes σ_B^2 also minimizes σ_W^2 . A simple approach to optimal thresholding is to perform an exhaustive sequential search for a threshold level t^* , which satisfies $\sigma_B^2(t^*) = \max_{1 \leq t < L} \sigma_B^2(t)$. This can be extended to n -level thresholding problem, which involves $n - 1$ thresholds that satisfy $\sigma_B^2(t_1^*, t_2^*, \dots, t_{n-1}^*) = \max_{1 \leq t_1 < t_2 < \dots < t_{n-1} < L} \sigma_B^2(t_1, t_2, \dots, t_{n-1})$.

The exhaustive search method based on the Otsu criterion is simple and straightforward, but it has a weakness that it is computationally expensive. The ranges of $n - 1$ candidate thresholds for n -level thresholding are as follows: $1 \leq t_1 < L - n + 1$, $t_1 + 1 \leq t_2 < L - n + 2$, and $t_{n-2} + 1 \leq t_{n-1} < L - 1$. Exhaustive search for $n - 1$ optimal thresholds involves evaluations of objective functions of $n(L - n + 1)^{n-1}$ combinations of thresholds. Therefore, it is not a suitable choice for the applications that require real-time multilevel image thresholding.

The task of determining $n - 1$ optimal thresholds for n -level image thresholding can be formulated as a multidimensional optimization problem. In this study, PSO and BFA have been used to determine the thresholds that maximize the between-class variance σ_B^2 of the intensity distributions. For PSO, the position of particle i is defined as $X_i = \{t_1, t_2, \dots, t_{n-1}\}$, and for BFA, the position of a bacterium i is defined as $P_i = \{t_1, t_2, \dots, t_{n-1}\}$. The particles in PSO and the bacteria in BFA are evaluated for the fitness function, which is defined as the between-class variance σ_B^2 of the image-intensity distributions. These are shown in (18) and (19), respectively

$$f(X_i) = \sigma_B^2(X_i) \quad (18)$$

$$J(P_i) = \sigma_B^2(P_i). \quad (19)$$

The goal of PSO is to determine the position in the search space that satisfies (20)

$$Xg_{\text{best}} = \max_{1 \leq t_1 < t_2 < \dots < t_{n-1} < L} \sigma_B^2(t_1, t_2, \dots, t_{n-1}). \quad (20)$$

Similarly, the goal of BFA is to determine the position in the search space that satisfies (21)

$$P |_{J_{\text{max}}} = \max_{1 \leq t_1 < t_2 < \dots < t_{n-1} < L} \sigma_B^2(t_1, t_2, \dots, t_{n-1}). \quad (21)$$

It is ensured that the thresholds determined by PSO and BFA are integer valued. This is done by rounding off the values in all dimensions of particle positions X_i in PSO and bacteria positions P_i in BFA. The optimal thresholds determined by PSO or BFA are used for segmentation of the image, and the results of segmentations are used for making a decision on sensor deployment.

IV. PSO- AND BFA-BASED ITERATIVE NODE LOCALIZATION

The objective of WSN node localization is to perform distributed estimation of coordinates of the maximum of N target nodes using M stationary beacons that know their locations. This study approaches node localization in a WSN in the following way.

- 1) N dumb nodes and M beacons are deployed from a UAV in a sensor field. Each dumb node and each beacon has a transmission radius of r units. Beacon nodes possess location awareness, and they frequently transmit their coordinates. The nodes that get settled at the end of an iteration serve as references in the next iteration, in which they transmit their location information as the beacons do.
- 2) Each node that falls within transmission radii of three or more noncollinear references (beacons or settled nodes) is referred to as a localizable node.
- 3) Each localizable node in the deployment measures its distance from each of its neighboring beacons or settled nodes. The effect of measurement noise is simulated as a Gaussian additive white noise. A node estimates its distance from a beacon i as $\hat{d}_i = d_i + n_i$, where d_i is the actual distance given by $d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$. Here, (x, y) is the location of the target node, and (x_i, y_i) is the location of the i th beacon in the neighborhood of the target node. The measurement noise n_i has a random value uniformly distributed in the range $d_i \pm d_i(P_n/100)$. It is clear that the result of localization depends on the value of P_n , the percentage noise that affects distance measurements.
- 4) Each localizable node independently runs a bioinspired optimization algorithm discussed earlier. Two case studies are conducted. In case study 1, each localizable node runs PSO to localize itself. In case study 2, each localizable node runs BFA to localize itself. Both PSO and BFA find the coordinates (x, y) that minimize the objective function, which represents the error defined in (22) as follows:

$$f(x, y) = \frac{1}{M} \sum_{i=1}^M \left(\sqrt{(x - x_i)^2 + (y - y_i)^2} - \hat{d}_i \right)^2 \quad (22)$$

where $M \geq 3$ is the number of beacons or settled nodes within the transmission radius of the target node.

- 5) PSO and BFA search for the values of (x, y) that minimize the error, therefore, the search space is 2-D.
- 6) After all the N_L localizable nodes determine their coordinates, the total localization error is computed as the mean of squares of distances between actual node locations (x_i, y_i) and the locations (\hat{x}_i, \hat{y}_i) , $i = 1, 2, \dots, N_L$, determined by PSO or BFA. This is computed, as (23), as follows:

$$E_l = \frac{1}{N_L} \sum_{i=1}^L ((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2) \quad (23)$$

- 7) Steps 2 to 6 are repeated until either all dumb nodes get localized or no more nodes can be localized. The performance of the localization algorithm is determined by the doublet (N_{N_L}, E_l) , where $N_{N_L} = N - N_L$ is the number of nodes that could not be localized. The lower the values of N_{N_L} and E_l , the better the performance is.

As the iterations progress, the number of localized nodes increases. This increases the number of references available for already localized nodes. A node that localizes using just three references in an iteration k may have more references in iteration $k + 1$. This decreases the probability of the flip ambiguity. On the other hand, If a node has more references in iteration $k + 1$ than in iteration k , the time required for localization increases. Authors observed from exhaustive experimentation that the maximum number of references can be safely restricted to six.

V. NUMERICAL SIMULATION AND RESULTS

PSO- and BFA-based image thresholding and iterative localization algorithms proposed in this paper are validated through numeric simulations in MATLAB on a computer having Intel Core Duo T2300 processor (1.66 GHz/2 MB L2 Cache) and 1 GB of memory. Simulation details and the results obtained are discussed in Section V-A and V-B. PSO has fewer parameters to choose, and there are guidelines on how to choose them. Therefore, PSO experiments are conducted first, and BFA experiments are conducted with parameters chosen on trial and error in order to achieve results closest to those of PSO.

A. Image Thresholding for Autonomous Deployment

PSO and BFA are used to compute the optimal values of the thresholds by maximizing the between-class variance of the distribution of intensity levels in the given image. The parameters for PSO and BFA algorithms are chosen as follows:

PSO:

- 1) population = 20;
- 2) iterations = 10;
- 3) acceleration constants $c_1 = c_2 = 2.0$;
- 4) inertia weight is decreased linearly from 0.9 in the first iteration to 0.4 in the last iteration;
- 5) limits on velocities: $v_{\max} = 10$ and $v_{\min} = -10$;
- 6) limits on particle positions: $X_{\min} = 0$ and $X_{\max} = 255$.

BFA:

- 1) population = 20;
- 2) number of chemotactic steps $N_c = 5$;



Fig. 2. Aerial image *lake* used for testing PSO- and BFA-based thresholding.

- 3) number of swims $N_s = 8$;
- 4) number of reproduction steps $N_{re} = 5$;
- 5) number of elimination-dispersion steps $N_{ed} = 5$;
- 6) fraction of bacteria that split in each reproduction step $S_r = 0.5$;
- 7) probability that a bacterium is eliminated in an elimination-dispersion round $p_{ed} = 0.1$;
- 8) cell-to-cell signaling is not used. Therefore, the depth and the width of attractants d_a and w_a , and the height and the width of repellents h_r and w_r are irrelevant.

Three case studies are conducted on the image *lake* shown in Fig. 2. This image has 256 intensity levels. Average times taken for PSO and BFA search in 20 trial runs are taken for comparison. The results obtained in PSO and BFA search are compared with those from exhaustive search algorithm. The exhaustive search method is deterministic; therefore, it is not repeated for statistical analysis.

1) *Case Study I (Two-Level Thresholding):* PSO and BFA are used to determine the optimal threshold level used to dichotomize the given image. The optimal thresholds determined by both PSO and BFA are identical. The output of optimal thresholding of the *lake* image is shown in Fig. 3(a).

2) *Case Study II (Three-Level Thresholding):* Case study II deals with three-level thresholding. Here, PSO and BFA are used to determine two optimal threshold levels t_1^* and t_2^* . Initially, the positions of candidate solutions (particles in PSO and bacteria in BFA) are randomly assigned as integers between 1 and 256, such that $1 < t_1 < t_2 < 256$. The optimal thresholds determined by both PSO and BFA are identical. The output of optimal thresholding of the *lake* image is shown in Fig. 3(b).

3) *Case Study III (Multilevel Thresholding):* Here, PSO and BFA are used to determine $n - 1$ optimal threshold levels. The number of PSO iterations is increased to 50. The results of exhaustive, PSO- and BFA-based search for four-level and five-level thresholding of the *lake* image are shown in Fig. 3(c) and (d), respectively.

A summary of results for two-, three-, four-, and five-level thresholding is presented in Table I. The computation time

TABLE I
SUMMARY OF RESULTS OF EXHAUSTIVE, PSO AND BFA SEARCH FOR OPTIMAL MULTILEVEL THRESHOLDING OF THE IMAGE LAKE

| Number of levels | Exhaustive search | | PSO search | | BFA search | |
|------------------|----------------------|-----------------------------|----------------------|-----------------------------|----------------------|-----------------------------|
| | Optimal threshold(s) | Computing time [†] | Optimal threshold(s) | Computing time [†] | Optimal threshold(s) | Computing time [†] |
| 2 | 112 | 0.25 s | 112 | 0.25 s | 112 | 5.17 s |
| 3 | 103,146 | 2.25 s | 103,146 | 0.52 s | 103,146 | 8.43 s |
| 4 | 94,125,155 | 147.00 s | 94,125,155 | 0.70 s | 94,125,155 | 10.48 s |
| 5 | 90,119,143,179 | 221.00 s | 90,119,143,179 | 0.84 s | 90,119,143,179 | 12.2 s |

[†]All experiments are conducted on the same computer.

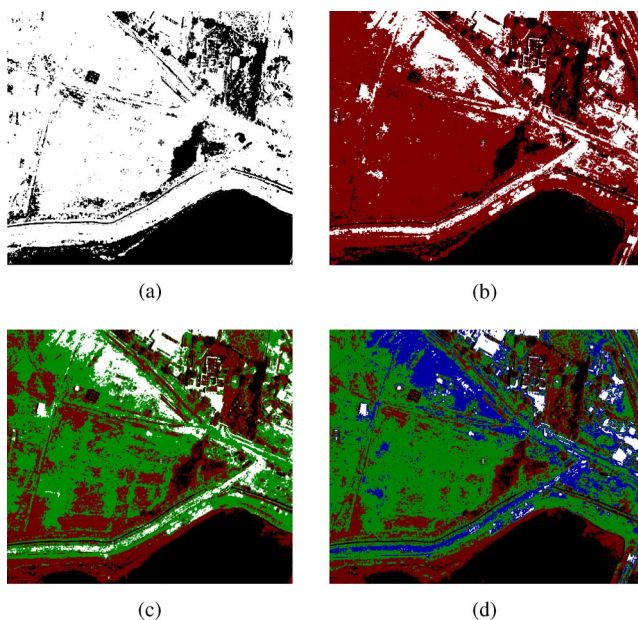


Fig. 3. Results of PSO- and BFA-based thresholding (gray levels are shown in red, green, and blue shades). (a) Two-level thresholding. (b) Three-level thresholding. (c) Four-level thresholding. (d) Five-level thresholding.

required for exhaustive, PSO- and BFA-based search methods for varying number of thresholding levels is shown in Fig. 4. It can be observed that the computational time for three-level exhaustive thresholding is significantly larger than that for four-level. This is attributed to the fact that the total number of candidate threshold combinations increases exponentially as the number of levels increases. It may be noted that in all trial runs, the optimal threshold levels determined by both PSO and BFA are identical to each other, and to those determined by the exhaustive search method. The results show that PSO is fastest among the studied algorithms in determining the optimal thresholds.

In autonomous deployment of WSN nodes, the image of the terrain taken from a downward-pointed camera is segmented using a bioinspired algorithm. The threshold information is used to decide if a node can be dropped at a location. For example, in the terrain that threshold versions of the image *lake* show water and the vegetated zones as black pixels. Deployment is avoided in such zones. Thirty trial runs of PSO- and BFA-based autonomous deployment are conducted on the terrain represented by the image *lake*. The objective here is to deploy the sensors on dry land and avoid sensor deployment in water. In each experiment, 40 nodes and eight beacons are deployed based on PSO- and BFA-based optimal thresholding. The results of one such experiment are shown in Fig. 5.

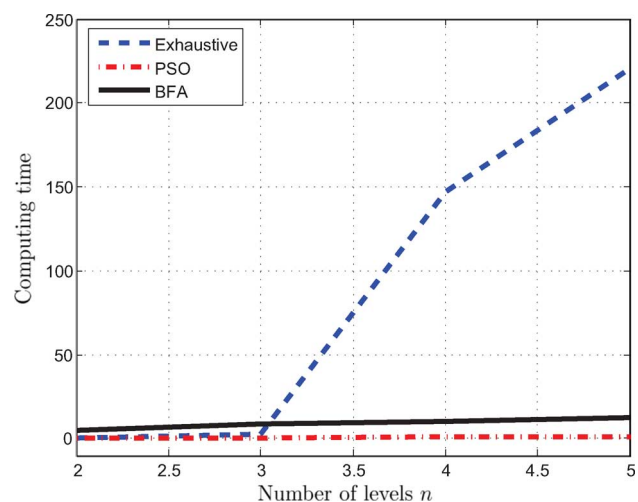


Fig. 4. Plot of number of thresholding levels versus the computational time.

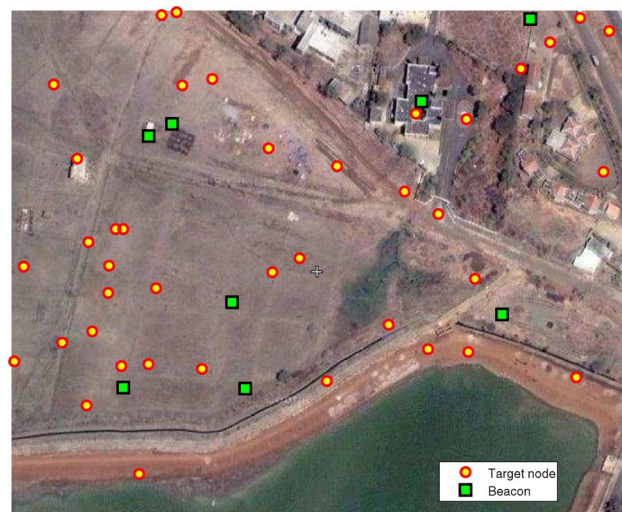


Fig. 5. Scenario of the WSN deployed based on thresholding the aerial image of the terrain.

The deployment scenario shows all sensor nodes and beacons deployed in the regions outside the lake. On average, the number of nodes saved from falling into the terrain of no interest (water in this case) is 8.99.

B. Iterative Node Localization

Simulation of the WSN and its performance evaluation is done in MATLAB. Forty target nodes and eight beacons are randomly deployed in a sensor field having dimensions of $100 \times$

100 square units. The numbers of target nodes and beacons and the size of the mission field are chosen arbitrarily. Each beacon has a transmission radius of $r = 30$ units. Simulation settings specific to case studies 1 and 2, and the result obtained are presented in the following sections.

1) *Case Study 1 (PSO-Based Localization)*: In this case study, each target node that can be localized runs a 2-D PSO to localize itself. Some PSO parameters are changed as follows:

- 1) population = 30;
- 2) iterations = 150;
- 3) limits on particle positions: $X_{\min} = 0$ and $X_{\max} = 100$.

Thirty trial experiments of PSO-based localization are conducted for $P_n = 2$ and $P_n = 5$. Average of total localization error E_l defined in (23) in each iteration in 30 runs is computed.

2) *Case Study 2 (BFA-Based Localization)*: In this case study, each localizable target node runs a 2-D BFA to localize itself. Some BFA parameters are changed as follows:

- 1) population = 30;
- 2) number of swims $N_s = 20$.

Thirty BFA-based localization experiments are conducted for $P_n = 2$ and $P_n = 5$. Both the algorithms studied here are stochastic; therefore, they do not produce the same solutions in all trials even with identical initial deployment. This is the reason why the results of multiple trial runs are averaged. Besides, initial deployment is random; therefore, the number of localizable nodes in each iteration is not the same. This affects the total computing time.

The actual locations of nodes and beacons, and the coordinates of the nodes estimated by PSO and BFA in a trial run are shown in Fig. 6. The initial deployment of nodes and beacons for PSO- and BFA-based localization is the same in a trial run. Results of PSO- and BFA-based localization summarized in Table II show that both stochastic algorithms used here have performed fairly well in WSN localization. The effect of P_n , percentage noise in distance measurement, on localization accuracy can be clearly seen. Average localization error in both PSO and BFA is reduced for $P_n = 2$. The performance metric doublet (N_{NL}, E_l) for BFA is less than that for PSO, thus indicating superior performance of BFA. However, computing time required for BFA is significantly more than that for PSO, which is a weakness of BFA. In addition, the amount of memory required for BFA is more than that for PSO. This clearly calls for a tradeoff. A choice between PSO and BFA is influenced by how constrained the nodes are in terms of memory and computing resources, how accurate the localization is expected to be, and how quickly that should happen.

The detailed observations made in the first five trial runs out of the 30 experiments are summarized in Table III. Table III depicts increasing N_L , the number of localized nodes, in each iteration. It also shows the correction of large errors due to flip ambiguity. It may be noted that in trial 2, all 40 nodes get localized at the end of the third iteration. In trial 4, only 39 nodes could be localized in the third iteration. The 40th node could not get three or more references in its transmission radius.

It is clear that P_n , the maximum amount of Gaussian additive noise associated with distance measurements, is an important parameter that influences the accuracy of localization. It is

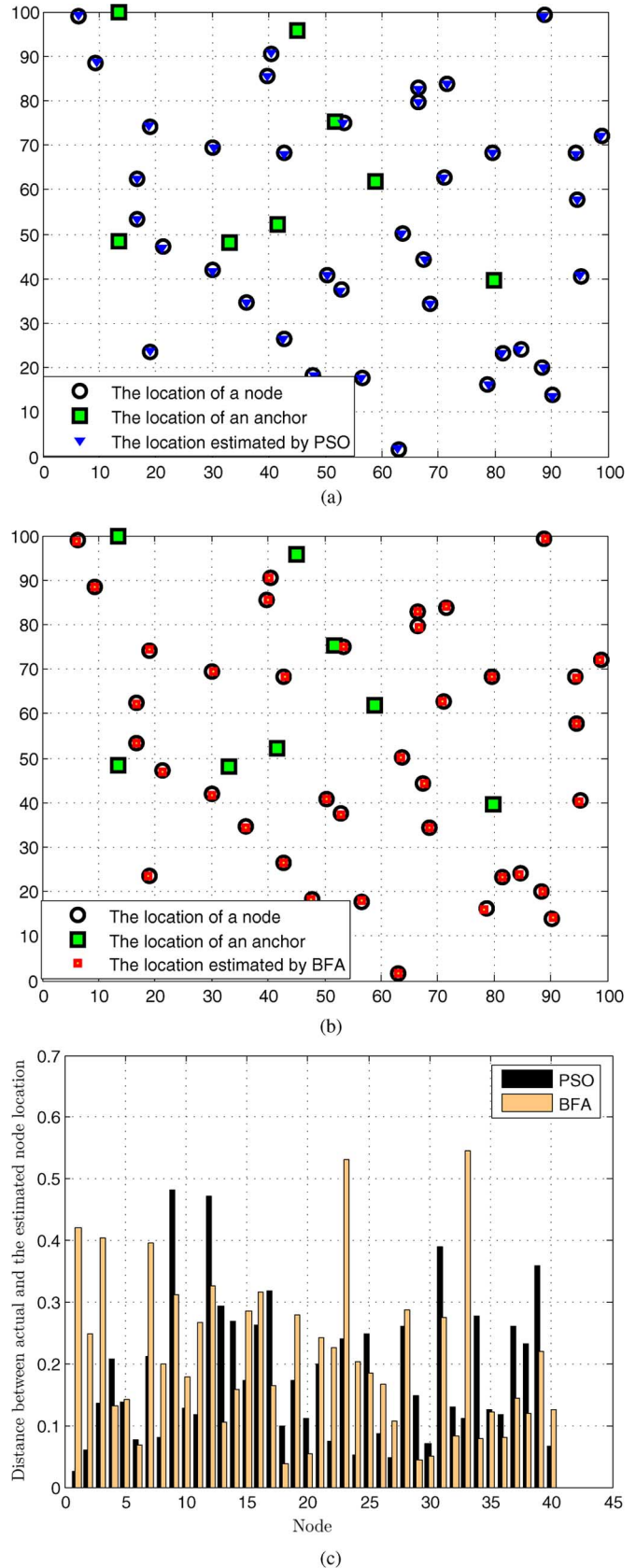


Fig. 6. Results of a trial run of PSO- and BFA-based iterative localization for $N = 40$, $M = 8$, $r = 30$, and $P_n = 2$ and the sensor field size of 100×100 . (a) Locations estimated by PSO. (b) Locations estimated by BFA. (c) Distances between actual locations and those estimated by PSO and BFA.

TABLE II
SUMMARY OF RESULTS OF 30 TRIAL RUNS PSO- AND BFA-BASED WSN NODE LOCATION

| | $P_n = 5$ | | | $P_n = 2$ | | |
|-----|---------------|---------------|---------------------|---------------|---------------|---------------------|
| | Mean N_{NL} | Mean E_l | Computing time* (s) | Mean N_{NL} | Mean E_l | Computing time* (s) |
| PSO | 1.666 | 0.2214 | 161.4267 | 1.5 | 0.0433 | 154.2232 |
| BFA | 1.333 | 0.1762 | 362.0218 | 0.6666 | 0.0298 | 334.8371 |

$N = 40$, $M = 8$, and $r = 30$ units and the sensor field size = 100×100 square units.
*All experiments are conducted on the same computer.

TABLE III
SUMMARY OF RESULTS OF PSO- AND BFA-BASED WSN NODE LOCATION WITH THE SAME INITIAL DEPLOYMENT OF NODES AND BEACONS WITH $N = 40$, $M = 8$, AND $r = 30$ UNITS, $P_n = 2$, AND THE SENSOR FIELD SIZE = 100×100 SQUARE UNITS

| Trial | | PSO | | | | BFA | | |
|-------|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 | Iteration 1 | Iteration 2 | Iteration 3 |
| 1 | N_L | 22 | 34 | 37 | 38 | 22 | 38 | 40 |
| | E_l | 0.0987 | 0.0492 | 0.00627 | 0.0538 | 0.0898 | 0.0919 | 0.0432 |
| | T_l | 8.469 | 30.8440 | 47.4370 | 65.8440 | 33.6870 | 118.2970 | 188.1090 |
| 2 | N_L | 20 | 36 | 40 | | 20 | 36 | 40 |
| | E_l | 158.4939* | 0.0611 | 0.0465 | | 16.3762 | 1.3977 | 0.0298 |
| | T_l | 7.1410 | 29.0150 | 47.2170 | | 29.0780 | 108.5150 | 174.3910 |
| 3 | N_L | 22 | 34 | 37 | 38 | 22 | 38 | 40 |
| | E_l | 0.0987 | 0.0492 | 0.0627 | 0.0538 | 0.9090 | 0.1269 | 0.0437 |
| | T_l | 8.4320 | 31.1410 | 48.9060 | 62.5470 | 34.2970 | 120.3280 | 191.7510 |
| 4 | N_L | 24 | 38 | 39 | | 24 | 39 | 40 |
| | E_l | 21.8450* | 0.1045 | 0.0520 | | 0.2640 | 0.1757 | 0.0422 |
| | T_l | 8.9070 | 31.3280 | 52.7870 | | 36.2660 | 115.9840 | 187.9530 |
| 5 | N_L | 22 | 34 | 37 | 38 | 22 | 38 | 40 |
| | E_l | 0.0987 | 0.0492 | 0.0627 | 0.0538 | 0.0912 | 0.1296 | 0.0395 |
| | T_l | 8.4060 | 31.2187 | 48.0470 | 62.0620 | 34.3470 | 120.7180 | 192.0940 |

All experiments are conducted on the same computer.
*These large errors due to flip ambiguities are reduced in subsequent iterations.

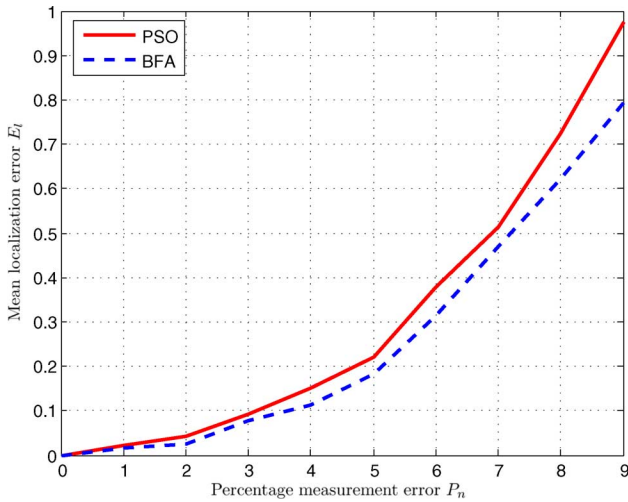


Fig. 7. Dependence of the localization error E_l on percentage measurement noise P_n .

expected that the accuracy decreases, thus leading to an increase in the localization error E_l as P_n increases. The dependence of E_l on P_n is studied over ten trial runs for each value of P_n between 0 and 9. Variation of E_l with respect to P_n is shown in Fig. 7. The localization methods studied here are iterative. The number of nodes localized N_L increases with iterations. Having more number of beacons is advantageous because it gives more number of references for dumb nodes. The percentage of nodes that get localized depends on the node density and the number of beacons. A summary of ten trial runs, each for the number of beacons M varying between 4 and 12, is shown in Fig. 8. This ob-

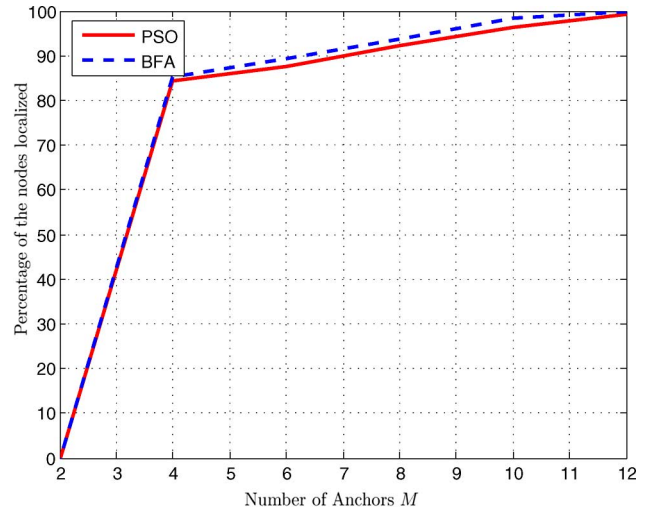


Fig. 8. Dependence of percentage of nodes localized on the number of the original anchor nodes (M).

servations also provides an insight into the cost tradeoff between a WSN with all $M + N$ nodes having on-board GPS devices and a WSN that has M beacons and N ordinary nodes. Summary of results of ten random deployments conducted with $N = 40$, $r = 30$ units, and $P_n = 2$ in the sensor field size = 100×100 square units shows that 12 GPS-enabled anchor nodes are necessary in order to localize all 40 dumb nodes. This results in about 77% saving in the cost of GPS hardware, which comes at a price of inaccurate localization. With 12 anchor nodes, the average localization error E_l of 40 nodes is 0.05412 with PSO-based localization, and 0.03976 with BFA-based localization.

TABLE IV
PROPAGATION OF LOCALIZATION ERRORS FOR $r = 25$ UNITS, $P_n = 5$, AND THE SENSOR FIELD SIZE = 100×100 SQUARE UNITS

| Iteration ▼ | Node ► | N_1 | N_2 | N_3 | N_4 |
|-------------|-----------------|----------------------|----------------------|----------------------|-----------------|
| 1 | Reference nodes | B_1, B_2, B_3 | | | |
| | Error E (PSO) | 0.2771 | | | |
| | Error E (BFA) | 0.2632 | | | |
| 2 | Reference nodes | | B_2, B_3, N_1 | | |
| | Error E (PSO) | | 0.4247 | | |
| | Error E (BFA) | | 0.4124 | | |
| 3 | Reference nodes | B_1, B_2, B_3, N_2 | | B_4, B_5, N_2 | |
| | Error E (PSO) | 0.3139 | | 0.5892 | |
| | Error E (BFA) | 0.2844 | | 0.5929 | |
| 4 | Reference nodes | | B_2, B_3, N_1, N_3 | | B_6, B_7, N_3 |
| | Error E (PSO) | | 0.4428 | | 0.6927 |
| | Error E (BFA) | | 0.4519 | | 0.6919 |
| 5 | Reference nodes | | | B_4, B_5, N_2, N_4 | |
| | Error E (PSO) | | | 0.6732 | |
| | Error E (BFA) | | | 0.6641 | |

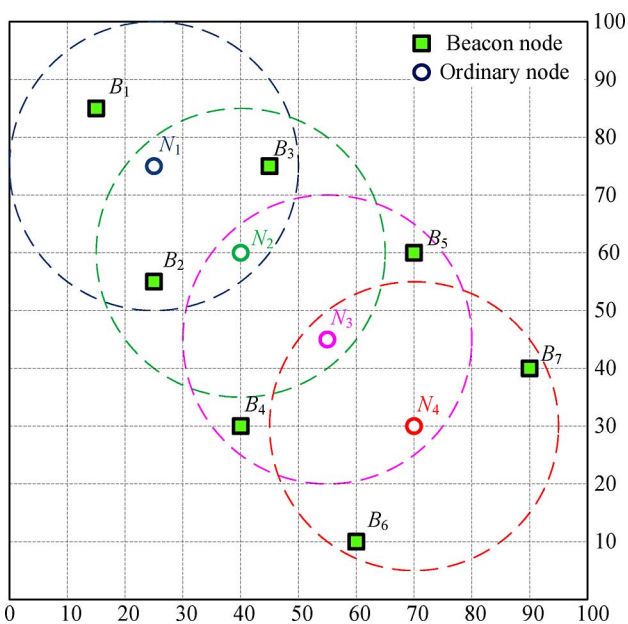


Fig. 9. Test deployment used to assess propagation of localization error.

3) *Error Propagation*: The localization methods discussed earlier suffer from the problem of error propagation. The test deployment of ordinary nodes N_1 through N_4 and beacons B_1 through B_7 , shown in Fig. 9, is used to assess the propagation of error, as more dumb nodes get settled and serve as beacons for other dumb nodes. In this deployment, positions of the nodes N_1 to N_4 and B_1 to B_7 are chosen in such a way that the nodes N_1, N_2, N_3 , and N_4 get localized in the first, the second, the third, and the fourth iteration, respectively. Besides, due to availability of additional pseudobeacons, nodes N_1, N_2 , and N_3 reestimate their locations in the third, the fourth, and the fifth iteration, respectively. Results are summarized in Table IV. The distance between the actual node location and the estimated location is shown as error E .

The results show that localization error E progressively increases as new nodes get settled and act as references for other dumb nodes. Besides due to an increased number of references, the localization error of settled nodes increases too (for example, the localization accuracy of N_1 can be observed to be decreased

from iteration 1 to iteration 3 because of the newly settled node N_2). However, presence of more than three references helps a node significantly in recovering from flip ambiguity. Error propagation limits the scalability of the WSN. A thorough analysis of propagation and control of localization error are out of scope of this paper. Error propagation in the algorithms discussed in this paper can be controlled using the following mechanisms [38].

- 1) *Error characterization*: Each node maintains its location estimate and location error, which it uses to characterize the error.
- 2) *Neighbor selection*: Each node excludes from its neighborhood the nodes that have high uncertainties.
- 3) *Update criterion*: Each node discards the new estimate of its locations if the error is larger than a preset threshold.

VI. CONCLUSION AND FUTURE WORK

Bioinspired algorithms PSO and BFA have been presented in this paper for segmentation of terrain images for autonomous deployment of WSN nodes from a UAV and for localization of the deployed nodes in a distributed and iterative fashion. Both tasks are treated as multidimensional optimization problems and solved using the aforementioned bioinspired algorithms. The algorithms have been briefly outlined and a statistical summary of their results is presented. Image-segmentation-based autonomous deployment presented in the paper reduces the number of sensor nodes deployed in the terrains of no interest. The algorithms proposed for multilevel thresholding are observed to be faster than the exhaustive search for optimal thresholds.

The distributed localization method proposed in this paper has the advantage of reduced number of transmissions to the base station, which helps the nodes to conserve their energy, which is a serious concern in most WSN applications. The results show that the proposed algorithms have a tradeoff issue. While PSO determines the node coordinates more quickly, BFA does so more accurately. A judicial choice between the algorithms depends on the desired localization accuracy and the desired quickness of localization.

This study can be extended in several directions. A vision system that considers the color and the texture of the terrains in the image may provide more accurate deployment, which can

be investigated. In a possible future extension, PSO and BFA can be applied for centralized localization in order to compare with the distributed localization method presented in this paper. Such a comparison with an emphasis on energy awareness is particularly useful. Besides, a comparison of the stochastic localization methods with the available deterministic methods can give a useful insight. The analysis presented shows that localization errors propagate as iterations progress. The control of error propagation is a potential and important direction further study.

REFERENCES

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
- [2] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Sampalli, and G. Sukhatme, "Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2004, vol. 4, pp. 3602–3608.
- [3] A. Ollero, M. Bernard, M. La Civita, L. van Hoesel, P. Marron, J. Lepley, and E. de Andres, "AWARE: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned aerial vehicles," in *Proc. IEEE Int. Workshop Saf., Secur. Rescue Robot. (SSRR)*, Sep. 2007, pp. 1–6.
- [4] A. Ollero and L. Merino, "Control and perception techniques for aerial robotics," *Annu. Rev. Control*, vol. 28, pp. 167–178, May 2004.
- [5] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *J. Electron. Imag.*, vol. 13, no. 1, pp. 146–168, Jan. 2004.
- [6] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.
- [7] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.
- [8] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur, "A theory of network localization," *IEEE Trans. Mobile Comput.*, vol. 5, no. 12, pp. 1663–1678, Dec. 2006.
- [9] A. Boukerche, H. A. B. Oliveira, E. F. Nakamura, and A. A. F. Loureiro, "Localization systems for wireless sensor networks," *IEEE Wireless Commun. Mag.*, vol. 14, no. 6, pp. 6–12, Dec. 2007.
- [10] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, pp. 57–66, Aug. 2001.
- [11] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Comput. Netw.*, vol. 51, no. 10, pp. 2529–2553, Jul. 2007.
- [12] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Nov. 25–29, 2001, vol. 5, pp. 2926–2931.
- [13] C. Savarese, J. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad hoc wireless sensor networks," in *Proc. USENIX Tech. Annu. Conf.*, Jun. 2002, pp. 317–327.
- [14] A. Savvides, H. Park, and M. B. Srivastava, "The N -hop multilateration primitive for node localization problems," *Mobile Netw. Appl.*, vol. 8, no. 4, pp. 443–451, Aug. 2003.
- [15] L. Doherty, K. S. J. Pister, and L. El Ghaoui, "Convex position estimation in wireless sensor networks," in *Proc. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, Apr. 22–26, 2001, vol. 3, pp. 1655–1663.
- [16] P. Biswas, T. C. Lian, T. C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Trans. Sens. Netw.*, vol. 2, no. 2, pp. 188–220, May 2006.
- [17] T. C. Liang, T. C. Wang, and Y. Ye, "A gradient search method to round the semidefinite programming relaxation solution for ad hoc wireless sensor network localization," Stanford Univ., Stanford, CA, Tech. Rep., Dec. 2004. Available: <http://www.stanford.edu/~yye/formal-report5.pdf>
- [18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Nov. 27–Dec. 1, 1995, vol. 4, pp. 1942–1948.
- [19] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, Jun. 2002.
- [20] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, and R. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.
- [21] R. Schaefer, *Foundations of Global Genetic Optimization*. New York: Springer-Verlag, 2007.
- [22] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization* (ser. Natural Computing Series). Berlin, Germany: Springer-Verlag, 2005.
- [23] Y. P. Chen, W. C. Peng, and M. C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 6, pp. 1460–1470, Dec. 2007.
- [24] B. Luitel and G. K. Venayagamoorthy, "Differential evolution particle swarm optimization for digital filter design," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 3954–3961.
- [25] D. H. Kim, A. Abraham, and J. H. Cho, "A hybrid genetic algorithm and bacterial foraging approach for global optimization," *Inf. Sci.*, vol. 177, no. 18, pp. 3918–3937, Sep. 2007.
- [26] H. Shen, Y. Zhu, X. Zhou, H. Guo, and C. Chang, "Bacterial foraging optimization algorithm with particle swarm optimization strategy for global numerical optimization," in *Proc. 1st ACM/SIGEVO Summit Genet. Evol. Comput.*, New York, NY: ACM, Jun. 2009, pp. 497–504.
- [27] A. Y. Saber and G. K. Venayagamoorthy, "Economic load dispatch using bacterial foraging technique with particle swarm optimization biased evolution," in *Proc. IEEE Swarm Intell. Symp.*, St. Louis, MO, Sep. 2008, pp. 1–8.
- [28] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligenc.* (ser. Artificial Intelligence). San Mateo, CA: Morgan Kaufmann, 2008.
- [29] G. K. Venayagamoorthy, "A successful interdisciplinary course on computational intelligence," *IEEE Comput. Intell. Mag.*, vol. 4, no. 1, pp. 14–23, Jan. 2009.
- [30] X. Hu, Y. Shi, and R. Eberhart, "Recent advances in particle swarm," in *Proc. Congr. Evol. Comput. (CEC)*, Jun. 19–23, 2004, vol. 1, pp. 90–97.
- [31] R. V. Kulkarni, G. K. Venayagamoorthy, A. Miller, and C. H. Dagli, "Network-centric localization in MANETs based on particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, St. Louis, MO, Oct. 2008, pp. 1–6.
- [32] A. Cervantes, I. M. Galvan, and P. Isasi, "AMPSO: A new particle swarm method for nearest neighborhood classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 5, pp. 1082–1091, Oct. 2009.
- [33] M. Donelli, R. Azaro, F. D. Natale, and A. Massa, "An innovative computational approach based on a particle swarm strategy for adaptive phased-arrays control," *IEEE Trans. Antennas Propag.*, vol. 54, no. 3, pp. 888–898, Mar. 2006.
- [34] H. Fan and Y. Shi, "Study on V_{max} of particle swarm optimization," in *Proc. Workshop Part. Swarm Optim.*, Indianapolis, IN: Purdue School of Engineering and Technology, Apr. 2001.
- [35] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. 7th Int. Conf. Evol. Program. (EP-VII)*, London, U.K.: Springer-Verlag, Mar. 1998, pp. 591–600.
- [36] T. K. Das, G. K. Venayagamoorthy, and U. O. Aliyu, "Bio-inspired algorithms for the design of multiple optimal power system stabilizers: SPPSO and BFA," *IEEE Trans. Ind. Appl.*, vol. 44, no. 5, pp. 1445–1457, Sep. 2008.
- [37] W. J. Tang, M. S. Li, Q. H. Wu, and J. R. Saunders, "Bacterial foraging algorithm for optimal power flow in dynamic environments," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 8, pp. 2433–2442, Sep. 2008.
- [38] J. Liu and Y. Zhang, "Error control in distributed node self-localization," *EURASIP J. Adv. Signal Process.*, vol. 2008, pp. 1–13, Jan. 2008.



Raghavendra V. Kulkarni (M'97–SM'05) received the M.Tech. degree in electronics engineering from the Institute of Technology, Banaras Hindu University, Varanasi, India, in 1994, and the Ph.D. degree in electrical engineering from Missouri University of Science and Technology, Rolla, in 2010.

His current research interests include the development of wireless sensor network applications using computational intelligence tools.

Dr. Kulkarni was the Registration and Publications Chair of the 2008 IEEE Swarm Intelligence Symposium. He is a Life Member of the Indian Society for Technical Education. He is a member of the IEEE Computational Intelligence Society and the International Neural Network Society.



Ganesh Kumar Venayagamoorthy (S'91–M'97–SM'02) received the Ph.D. degree in electrical engineering from the University of Natal, Durban, South Africa, in 2002.

In 2007, he was a Visiting Researcher at ABB Corporate Research, Sweden. He is currently an Associate Professor of electrical and computer engineering, and the Founder and the Director of the Real-Time Power and Intelligent Systems Laboratory, Missouri University of Science and Technology, Rolla. He was a Guest Editor of the *Neural Networks* journal. He has authored or coauthored two edited books, five book chapters, and more than 85 refereed journals papers and 275 refereed conference proceeding papers. He has received approximately US\$ 7 million of competitive research funding. His research interests include development and applications of advanced computational algorithms for real-world applications, including power systems stability and control, smart grid applications, sensor networks, and signal processing.

Dr. Venayagamoorthy is a Fellow of the Institution of Engineering and Technology, U.K. and the South African Institute of Electrical Engineers. He is a Senior Member of the International Neural Network Society (INNS) and a member of the American Society for Engineering Education. He is the Member of Board of Governors of INNS. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and an Editor of the IEEE TRANSACTIONS ON SMART GRID. He was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS from 2004 to 2007 and the IEEE TRANSACTIONS ON MEASUREMENTS AND INSTRUMENTATION in 2007. He was the recipient of several awards, including the 2007 U.S. Office of Naval Research Young Investigator Program Award, the 2004 National Science Foundation CAREER Award, and the 2010 Innovation Award from St. Louis Academy of Science.