Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 871596, 11 pages
doi:10.1155/2012/871596

Research Article

Linear Decision Fusion under the Control of Constrained PSO for WSNs

Sisi Jiang,[1] Zhiwen Zhao,[1,2] Sheng Mou,[1] Zushun Wu,[1] and Yi Luo[1]

[1] *College of Information Science and Technology, Beijing Normal University, Beijing 100875, China*
[2] *Key Laboratory of Inertial Technology for National Defense, Beihang University, Beijing 100191, China*

Correspondence should be addressed to Zhiwen Zhao, zhaozw126@126.com

Received 31 May 2011; Revised 8 September 2011; Accepted 26 September 2011

Academic Editor: Don Sofge

A major application of a distributed WSN (wireless sensor network) is to monitor a specific area for detecting some events such as disasters and enemies. In order to achieve this objective, each sensor in the network is required to collect local observations which are probably corrupted by noise, make a local decision regarding the presence or absence of an event, and then send its local decision to a fusion center. After that, the fusion center makes the final decision depending on these local decisions and a decision fusion rule, so an efficient decision fusion rule is extremely critical. It is obvious that the decision-making capability of each node is different owing to the dissimilar signal noise ratios and some other factors, so it is easy to understand that a specific sensor contribution to the global decision should be constrained by this sensor decision-making capability, and, based on this idea, we establish a novel linear decision fusion model for WSNs. Moreover, the constrained particle swarm optimization (constrained PSO) algorithm is creatively employed to control the parameters of this model in this paper and we also apply the typical penalty function to solve the constrained PSO problem. The emulation results indicate that our design is capable of achieving very high accuracy.

1. Introduction

Owing to the low cost and ease of operation, wireless sensor networks are ideal for a wide variety of applications such as environmental monitoring, smart factory instrumentation, intelligent transportation, and remote surveillance [1–3]. When a WSN is used to monitor an area, each sensor node in the network should collect local observations and send a summary (compressed or partially processed data) to the fusion center. Then the fusion center uses these summaries and a specific decision fusion rule to make the final global decision, as shown in Figure 1.

Actually, WSN issues such as node deployment, localization, energy-aware clustering, and data aggregation are always considered as optimization problems. An optimization method that requires moderate memory and computational resources and yet produces good results is desirable, especially for implementation on an individual sensor node. Bio-inspired optimization methods are computationally efficient alternatives to analytical methods. Particle swarm optimization (PSO) [4] is a popular multidimensional optimization technique. Ease of implementation, high quality of solutions, computational efficiency, and speed of convergence are the advantages of the PSO. In literatures [5–9], PSO is used to discover the optimal WSN deployment. Static deployment is a one-time process in which solution quality is more important than fast convergence. PSO suits centralized deployment. Fast PSO variants are necessarily of dynamic deployment, so PSO limits network scalability. Literatures [10–14] apply PSO for WSN localization. Owing to energy issues, the distributed localization is desirable. Though PSO is appropriate for distributed localization, the choice is influenced by availability of memory on the nodes. Besides, PSO is also used for energy-aware clustering in literatures [15–17], where clustering is a centralized optimization carried out in a resource-rich base station and optimal clustering has a strong influence on the performance of WSNs. Recently, some researchers have begun to use PSO for WSN data aggregation. As discussed in literature [18], PSO has provided optimization in several aspects of data aggregation as follows.

(a) In [19], Wimalajeewa and Jayaweera address the problem of optimal power allocation through constrained PSO. Their algorithm *PSO-Opt-Alloc* uses PSO to determine optimal-power allocation in the cases of both independent and correlated observations. The objective is to minimize the energy expenditure while keeping the fusion-error probability under a required threshold.

(b) In [20], Veeramachaneni and Osadciw present a hybrid of ant-based control and PSO *(ABC-PSO)* for hierarchy and threshold management for decentralized serial sensor networks. PSO is used to determine the optimal thresholds and fusion rules for the sensors while the ant colony optimization algorithm determines the hierarchy of sensor decision communication.

(c) In [21], Veeramachaneni and Osadciw present a binary multiobjective PSO *BMPSO* for optimal configuration for multiple sensor networks. PSO is modified to optimize two objectives: accuracy and time. The output of the algorithm is the choice of sensors, individual sensor threshold, and the optimal decision fusion rule.

As mentioned above, data fusion is a distributed repetitive process which is moderately suitable for PSO. Effective data fusion influences overall WSN performance and demands quick-convergence optimization techniques that assure high-quality solutions. Therefore, it is reasonable for us to choose PSO to control the parameters of our fusion rule.

In this paper, we design a linear decision fusion rule. In this rule, a linear equation is used to compute the integrated contribution of all the local decisions, and this equation is made up with local decision weight k_i and all local decisions. Local decision weight k_i can reflect a sensor decision-making capability, and the result of this equation is the summation of all the local sensor contributions. By comparing the result of the linear equation with a threshold λ_g, the final decision can be made in the fusion center. In order to get the smallest error probability, constrained PSO is employed to find out the optimal local decision weight k_i and the threshold λ_g.

The rest of the paper is organized as follows. Section 2 discusses the constrained-PSO algorithm briefly. Section 3 describes our linear decision fusion rule in detail. The performance of our fusion rule is evaluated in Section 4, and Section 5 concludes this paper.

2. Brief Introduction of Constrained PSO Algorithm

The particle swarm optimization (PSO) algorithm is a population-based evolutionary computation originally developed by Kennedy and Eberhart [4]. This algorithm does very well in searching throughout a highly multiple modal search space resulting in an optimized solution to an n-dimensional problem. PSO, originally introduced in terms of social and cognitive behavior, is widely used as a bottom-up problem solving method in engineering and computer science.

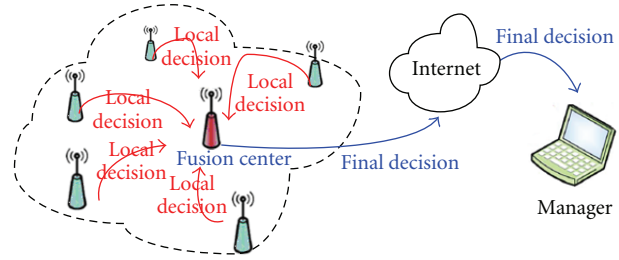


Figure 1: A typical structure of a decentralized decision fusion system.

In the basic PSO algorithm, a swarm consists of m particles flying around in an n-dimensional search space. The position of the ith particle at the tth iterations is used to evaluate the particle and represent the candidate solution for the optimization problem. It can be presented as $X_i^t = [x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t]$, where x_{ij}^t is the position value of the ith particle with respect to the jth dimension ($j = 1, 2, \ldots, n$). During the searching process, the position of a particle is affected by two factors: the best position visited by itself (p_{best}) denoted as $P_i^t = [p_{i1}^t, p_{i2}^t, \ldots, p_{in}^t]$ and the best position of the swarm found so far (g_{best}) denoted as $G^t = [g_1^t, g_2^t, \ldots, g_n^t]$. The new velocity (denoted as $V_i^t = [v_{i1}^t, v_{i2}^t, \ldots, v_{in}^t]$) and the position of particle i at the next iteration are calculated as

$$v_{ij}^{t+1} = w \cdot v_{ij}^t + c_1 r_1 \cdot \left(p_{ij}^t - x_{ij}^t\right) + c_2 r_2 \cdot \left(g_{ij}^t - x_{ij}^t\right),$$
$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1}, \tag{1}$$

where w is called the inertia parameter, c_1 and c_2 are, respectively, cognitive and social learning parameters, and r_1, r_2 are random numbers between (0,1). Relying on expressions (1), the particles can fly throughout search space toward p_{best} and g_{best} in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optimal. The process for a particle to update its location is as shown in Figure 2.

Further, a constrained PSO problem can be described as follows:

$$\min_x f(x) \quad (x \in R^n),$$
$$g_j(x) \geq 0 \quad (j = 1, 2, \ldots, q), \tag{2}$$
$$h_p(x) = 0 \quad (p = 1, 2, \ldots, m),$$

where $f(x)$ is the objective function, $g_j(x) \geq 0$ is the jth nonlinear constraint, and $h_p(x) = 0$ is the pth linear constraint. In this paper, we just discuss the situation with the linear constraint. The most common approach for solving constrained PSO problems is to take advantage of a penalty function. The constrained problem can be transformed into an unconstrained one by penalizing the constraints and building a single objective function, which in turn is minimized by using an unconstrained optimization algorithm [22–24].

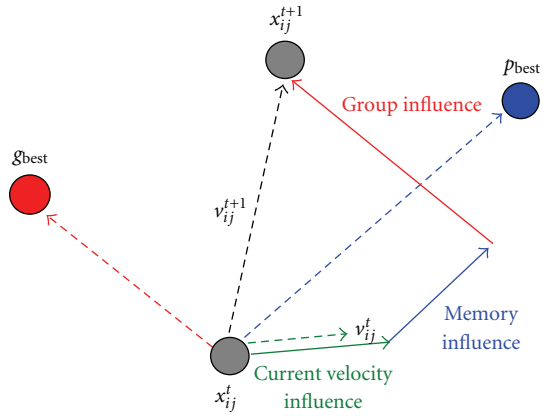


FIGURE 2: The process of updating location for each iteration in PSO.

Generally, penalty function can be classified into two main classes: stationary and nonstationary. Stationary penalty functions use fixed penalty values throughout the minimization, while in contrast, in nonstationary penalty functions, the penalty values are dynamically modified. In this paper, we just use the stationary penalty function as expression (3), which is generally defined as [25]

$$F(x) = f(x) + MH(x), \quad x \in R^n, \tag{3}$$

where $f(x)$ is the original objective function of the constrained PSO problem, M is a fixed penalty value, and $H(x)$ is a penalty factor. In this way, the constrained problem becomes unconstrained and then we can use the basic PSO to find out the smallest $F(x)$.

Depending on the knowledge about the constrained PSO, we would employ it to find out the optimal solution for the parameters of our decision fusion rule. In the next section, we would like to discuss this issue in detail.

3. The Linear Decision Fusion Rule

In this section, we plan to discuss this topic in three parts. In Section 3.1, we formulate the decision fusion problem regarding a binary hypothesis situation and the expressions of some crucial parameters are discussed, including the local error probabilities α_i, β_i, and threshold λ_i. In Section 3.2, our linear decision fusion rule is introduced and, with the expressions given by Section 3.1, the expressions of global error probabilities α_g, β_g, and total error probability R can be figured out. From Section 3.2, it can be seen that the total error probability R mainly depends on weights $k_1, k_2, \ldots, k_n$ and λ_g, so we would like to discuss how to use PSO to optimize the parameters $(k_1, k_2, \ldots, k_n, \lambda_g)$ for our fusion rule in Section 3.3.

3.1. Decision Fusion Problem Formulation. We consider a binary hypothesis testing problem in an n-node distributed wireless network. The ith sensor observation under the two hypotheses is given by

$$\begin{aligned} H_0 &: z_i = v_i, \qquad u_i = 0, \quad i = 1, 2, \ldots, n \\ H_1 &: z_i = x_i + v_i, \qquad u_i = 1, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{4}$$

where v_i is zero-mean Gaussian observation noise with variance σ_i^2, x_i is the signal to be detected, and u_i is the decision to be made. In this paper, we consider the detection of a constant signal (e.g., $x_i = m$). As a brief review, the problem of radar detection can be formulated as a hypothesis testing problem where the two hypotheses are

$$u_i = \begin{cases} 0, & \text{the enemy does not exist in the network,} \\ 1, & \text{the enemy exists in the network.} \end{cases} \tag{5}$$

The conditional probability density functions are $p(z_i \mid H_1)$ and $p(z_i \mid H_0)$, so the decision could be made based on the following likelihood ratio test:

$$u_i = \begin{cases} 1, & \text{when } \dfrac{p(z_i \mid H_1)}{p(z_i \mid H_0)} > l_0, \\ 0, & \text{when } \dfrac{p(z_i \mid H_1)}{p(z_i \mid H_0)} < l_0, \end{cases} \tag{6}$$

where l_0 is a threshold computed according to the Bayesian framework as follows:

$$l_0 = \frac{(C_{10} - C_{00})q}{(C_{01} - C_{11})p}. \tag{7}$$

C_{ij} is the error cost (C_{10}: we consider there is no enemy but actually the enemy exists in the network; C_{01}: we consider the enemy exists but actually there is no enemy in the network; C_{00} and C_{11} are the costs when we judge the existence of the enemy rightly), and the prior probabilities of the two hypotheses H_1 and H_0 are denoted by $P(H_1) = p$ and $P(H_0) = q$, respectively. In real applications, the value of C_{10} and C_{01} depends on what kinds of error would cause more serious effect. Taking the radar detection mentioned above as an example, in the real world the error that the radar misses an enemy is much more serious than the error that the radar considers something else as an enemy, so in this example $C_{10} > C_{01}$. In this paper, we set $C_{10} = 0.8$, $C_{01} = 0.2$, and $C_{00} = C_{11} = 0$, so expression (7) becomes

$$l_0 = \frac{C_{10}q}{C_{01}p}. \tag{8}$$

The values of prior probabilities p and q depend on prior experiences, so we assume that they can be known before applying this decision fusion rule. Let us consider the detection of a constant signal $x_i = m$, and the noise v_i mentioned above is zero-mean Gaussian observation noise with variance σ_i^2, so, for all i, the ith sensor observation obeys Gaussian distribution as

$$\begin{aligned} H_0 &: z_i \sim N(0, \sigma_i^2), \quad i = 1, 2, \ldots, n, \\ H_1 &: z_i \sim N(m, \sigma_i^2), \quad i = 1, 2, \ldots, n. \end{aligned} \tag{9}$$

Consequently, the conditional probability density functions $p(z_i \mid H_1)$ and $p(z_i \mid H_0)$ can be calculated as

$$p(z_i \mid H_1) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(x-m)^2}{2\sigma_i^2}\right],$$
$$p(z_i \mid H_0) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{x^2}{2\sigma_i^2}\right], \tag{10}$$

and then, with (6) and (10), the decision rule of the ith sensor can be formulated as

$$u_i = \begin{cases} 1, & \text{when } x_i > \dfrac{m}{2} + \sigma_i^2 \ln \dfrac{C_{10}q}{C_{01}p}, \\ 0, & \text{when } x_i > \dfrac{m}{2} + \sigma_i^2 \ln \dfrac{C_{10}q}{C_{01}p}. \end{cases} \tag{11}$$

According to (11), the threshold of the ith sensor is

$$\lambda_i = \frac{m}{2} + \sigma_i^2 \ln \frac{C_{10}q}{C_{01}p}. \tag{12}$$

From (12), we can see that the threshold of each sensor is different owing to the different σ_i which represents the noise character of a sensor. Accordingly, taking advantage of (10), (12), and (13), the local error probabilities α_i and β_i can be obtained as follows:

$$\alpha_i = P(u_i = 1 \mid H_0) = \int_{\lambda_i}^{\infty} p(z_i \mid H_0)dz_i,$$
$$\beta_i = P(u_i = 0 \mid H_1) = \int_{0}^{\lambda_i} p(z_i \mid H_1)dz_i. \tag{13}$$

3.2. Our Linear Decision Fusion Rule. In this paper, we design a novel decision fusion model and in this model, a linear equation is used to denote the integrated contribution of all the local decisions as

$$z_g = \sum_{i=1}^{n} k_i u_i, \tag{14}$$

where u_i is the local decision of the ith sensor and k_i is the corresponding weight which is able to indicate the ith sensor decision-making capability. In order to understand this model better, we take a group of workers who are working together to check an event for example. Firstly, all the workers are required to try their best to draw a conclusion for this event, and then they send their decision to their leader. The leader makes a final decision depending on all the conclusions given by the workers. During this decision-making process, the decisions made by the more capable workers would be paid more attention by the leader. In this example, the workers correspond to the sensors in the mobile sensor networks, the leader corresponds to the fusion center, and the ability of the workers corresponds to the weight k_i. When a sensor decision-making capability is better, its weight could be greater. In other words, if a sensor can make a decision more accurately, its contribution to z_g is considered to be bigger, so it is reasonable to use $z_g = \sum_{i=1}^{n} k_i u_i$ to denote the integrated contribution of local decisions in the fusion center. Then, z_g would be compared with a threshold λ_g, and the final decision can be made according to the following inequality expression:

$$u_g = \begin{cases} 1, & \text{when } z_g > \lambda_g, \\ 0, & \text{when } z_g < \lambda_g. \end{cases} \tag{15}$$

Suppose all the local decisions are independent of each other; then we can get $(P(u_1u_2\cdots u_n \mid H_0) = \prod_{i=1}^{n} P(u_i \mid H_0)$ and $(P(u_1u_2\cdots u_n \mid H_1) = \prod_{i=1}^{n} P(u_i \mid H_1)$. Besides, the error probabilities α_g and β_g take all the combinations of the local decisions (i.e., $u_1u_2\cdots u_n = 00\cdots0 \sim 11\cdots1$) into account. Therefore, the two kinds of errors that the fusion center may make can be formulated as follows:

$$\begin{aligned} \alpha_g &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} P_{u_1u_2\cdots u_n}\left(u_g = 1 \mid H_0\right) \\ &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} P_{u_1u_2\cdots u_n}\left(\sum_{i=1}^{n} k_i u_i > \lambda_g \mid H_0\right) \\ &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} \left(P(u_1u_2\cdots u_n \mid H_0) \times P\left(\sum_{i=1}^{n} k_i u_i > \lambda_g\right)\right) \\ &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} \left(\prod_{i=1}^{n} P(u_i \mid H_0) \times P\left(\sum_{i=1}^{n} k_i u_i > \lambda_g\right)\right), \\ \beta_g &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} P_{u_1u_2\cdots u_n}\left(u_g = 0 \mid H_1\right) \\ &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} P_{u_1u_2\cdots u_n}\left(\sum_{i=1}^{n} k_i u_i < \lambda_g \mid H_1\right) \\ &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} \left(P(u_1u_2\cdots u_n \mid H_1) \times P\left(\sum_{i=1}^{n} k_i u_i < \lambda_g\right)\right) \\ &= \sum_{u_1u_2\cdots u_n=00\cdots0}^{11\cdots1} \left(\prod_{i=1}^{n} P(u_i \mid H_1) \times P\left(\sum_{i=1}^{n} k_i u_i < \lambda_g\right)\right), \end{aligned} \tag{16}$$

where

$$\begin{aligned} P(u_i = 0 \mid H_0) &= q - \alpha_i, \\ P(u_i = 1 \mid H_0) &= \alpha_i, \\ P(u_i = 0 \mid H_1) &= \beta_i, \\ P(u_i = 1 \mid H_1) &= p - \beta_i. \end{aligned} \tag{17}$$

According to (13), (16), and (17), the total error cost can be calculated as

$$R = \frac{C_{10}q}{C_{01}p}\alpha_g + \beta_g. \tag{18}$$

In order to understand this model better, we take the situation $n = 2$ for example. Suppose there are only two sensors in the network and their local decisions are u_1 and u_2, respectively. Then according to the linear decision fusion model, the integrated contribution of the two local decisions can be figured out as $z_g = k_1u_1 + k_2u_2$.

It is obvious that there are four combinations of u_1, u_2, and z_g, as shown in Table 1.

Then referring to expression (16), we can have

$$\begin{aligned}
\alpha_g &= P\left(u_g = 1 \mid H_0\right) \\
&= P(u_1 = 0 \mid H_0)P(u_2 = 0 \mid H_0)P\left(0 > \lambda_g\right) \\
&\quad + P(u_1 = 1 \mid H_0)P(u_2 = 0 \mid H_0)P\left(k_1 > \lambda_g\right) \\
&\quad + P(u_1 = 0 \mid H_0)P(u_2 = 1 \mid H_0)P\left(k_2 > \lambda_g\right) \\
&\quad + P(u_1 = 1 \mid H_0)P(u_2 = 1 \mid H_0)P\left(k_1 + k_2 > \lambda_g\right), \\
P(u_1 = 0 \mid H_0) &= q - \alpha_1, \\
P(u_1 = 1 \mid H_0) &= \alpha_1, \\
P(u_2 = 0 \mid H_0) &= q - \alpha_2, \\
P(u_2 = 1 \mid H_0) &= \alpha_2, \\
\beta_g &= P\left(u_g = 0 \mid H_1\right) \\
&= P(u_1 = 0 \mid H_1)P(u_2 = 0 \mid H_1)P\left(0 < \lambda_g\right) \\
&\quad + P(u_1 = 1 \mid H_1)P(u_2 = 0 \mid H_1)P\left(k_1 < \lambda_g\right) \\
&\quad + P(u_1 = 0 \mid H_1)P(u_2 = 1 \mid H_1)P\left(k_2 < \lambda_g\right) \\
&\quad + P(u_1 = 1 \mid H_1)P(u_2 = 1 \mid H_1)P\left(k_1 + k_2 < \lambda_g\right), \\
P(u_1 = 0 \mid H_1) &= \beta_1, \\
P(u_1 = 1 \mid H_1) &= p - \beta_1, \\
P(u_2 = 0 \mid H_1) &= \beta_2, \\
P(u_2 = 1 \mid H_1) &= p - \beta_2,
\end{aligned} \tag{19}$$

and then we can achieve

$$R = \frac{C_{10}q}{C_{01}p}\alpha_g + \beta_g. \tag{20}$$

Here, it can be seen that the value of the total error probability R depends upon k_1, k_2, and λ_g, so if the optimal k_1, k_2, and λ_g can be found, the smallest error probability R can be achieved. In the following content, let us discuss how to get the optimal weights k_i and threshold λ_g with the help of constrained PSO.

Table 1: The combinations of u_1, u_2, and z_g.

u_2	u_1	z_g
0	0	0
0	1	k_1
1	0	k_2
1	1	$k_1 + k_2$

Table 2: PSO terminology and corresponding parameters of linear decision fusion rule.

PSO terminology	Linear decision fusion rule
Location	$(k_1, k_2, \ldots, k_n, \lambda_g)$
Fitness	$R' = (C_{10}q/C_{01}p)\alpha_g + \beta_g + M(\sum_{i=1}^{n} k_i - 1)^2$ $(k_i \geq 0)$
p_{best}	The location in parameter space of the smallest R′ returned for a specific particle
g_{best}	The location in parameter space of the smallest R′ returned in the entire swarm
$V_{\max}$	The maximum allowed velocity range in a given direction $[0, 1]$
$X_{\max}$	The maximum allowed range in a given direction $[0, 1]$

3.3. Using Constrained PSO to Control the Linear Decision Fusion Rule. It is obvious that if all the local decisions are $H_1(u_i = 1, i = 1, 2, \ldots, n)$, the final decision must be $H_1(z_g = 1)$, so we can consider that $\sum_{i=1}^{n} k_i = 1$ $(k_i \geq 0)$, and then, it becomes a constrained PSO problem which can be denoted as follows:

$$\begin{aligned}
R &= \frac{C_{10}q}{C_{01}p}\alpha_g + \beta_g, \\
\sum_{i=1}^{n} k_i &= 1 \quad (k_i \geq 0).
\end{aligned} \tag{21}$$

Here, $R = (C_{10}q/C_{01}p)\alpha_g + \beta_g$ is the initial objective function and $\sum_{i=1}^{n} k_i = 1$ $(k_i \geq 0)$ is the linear constraint. The penalty function approach is used to solve this problem. According to formulation (3) the penalty function can be established as

$$R' = \frac{C_{10}q}{C_{01}p}\alpha_g + \beta_g + M\left(\sum_{i=1}^{n} k_i - 1\right)^2, \tag{22}$$

where M is the fixed penalty value and $(\sum_{i=1}^{n} k_i - 1)^2$ is the penalty factor. Then we should try to find out the smallest R′ taking advantage of the basic PSO.

Basing on the analysis above, we can map from the basic PSO to the linear decision fusion rule, as shown in Table 2.

After that, the PSO is carried out as follows.

(1) Firstly, the locations of all the particles are initialized and the current location is considered as p_{best}.

(2) Secondly, the velocity and the location of each node are updated in accordance with expressions (1) and (3).

Table 3: The simulation results with different inertia parameter w when the sensor number is 10.

Inputs ($c_1 = c_2 = 2$)	Outputs	
Inertia parameter w	Total error probability R	Time consumed for convergence (iterations)
$w = 0.5$	0.00181079	447
$w = w_{\max} - ((w_{\max} - w_{\min})/T_{\max}) \times t = 0.9 - (0.5/1500) \times \mathrm{t}$	0.00120768	123

Table 4: The simulation results with different learning parameters c_1 and c_2 when the sensor number is 10.

Inputs ($w = w_{\max} - ((w_{\max} - w_{\min})/T_{\max}) \times t$)	Outputs	
c_1 and c_2	Total error probability R	The number of iteration
$c_1 = c_2 = 2$	0.00120768	123
$c_1 = 0, c_2 = 2$	0.00274277	150
$c_1 = 2, c_2 = 0$	0.00121506	Cannot converge

Table 5: The inputs and outputs of the simulation.

Inputs (mean = 0, $M = 10$, $P = 0.8$, $q = 0.2$)			Outputs	
n	Differences	$(k_1, k_2, \ldots, k_n)$	λ_g	$R = R' - M(\sum_{i=1}^{n} k_i - m)^2 k_i$
5	(43.86, 52.63, 26.2, 39.39, 80.08)	(0.102567, 0.463625, 0.0806701, 0.019778, 0.333359)	0.704058	0.0342967
10	(43.86, 52.63, 26.2, 39.39, 80.08, 47.16, 45.09, 34.65, 67.89, 54.78)	(0.191843, 0.253623, 0.0650721, 0.00822591, 0, 0.158283, 0.0213777, 0.251185, 0.00907494, 0.0413086)	0.745094	0.00116502
15	(43.86, 52.63, 26.2, 39.39, 80.08, 47.16, 45.09, 34.65, 67.89, 54.78, 34.76, 56.34, 27.89, 47.89, 38.97)	(0.0244901, 0.157784, 0.368528, 0.10523, 0.0614695, 0.0173535, 0.0114619, 0, 0.00495967, 0.048092, 0.00231704, 0, 0.0210975, 0.127067, 0.0500054)	0.334601	0.0000403

(3) Thirdly, the new fitness is computed and the new p_{best} and g_{best} are generated.

(4) Finally, if the number of iterations increases to $T_{\max}$, then stop; otherwise, proceed back to step two. In this way, the optimal $(k_1, k_2, \ldots, k_n, \lambda_g)$ is discovered and g_{best} is the smallest R'. Therefore, the smallest total error probability can be computed as $R = R' - M(\sum_{i=1}^{n} k_i - 1)^2$.

4. Performance Evaluation

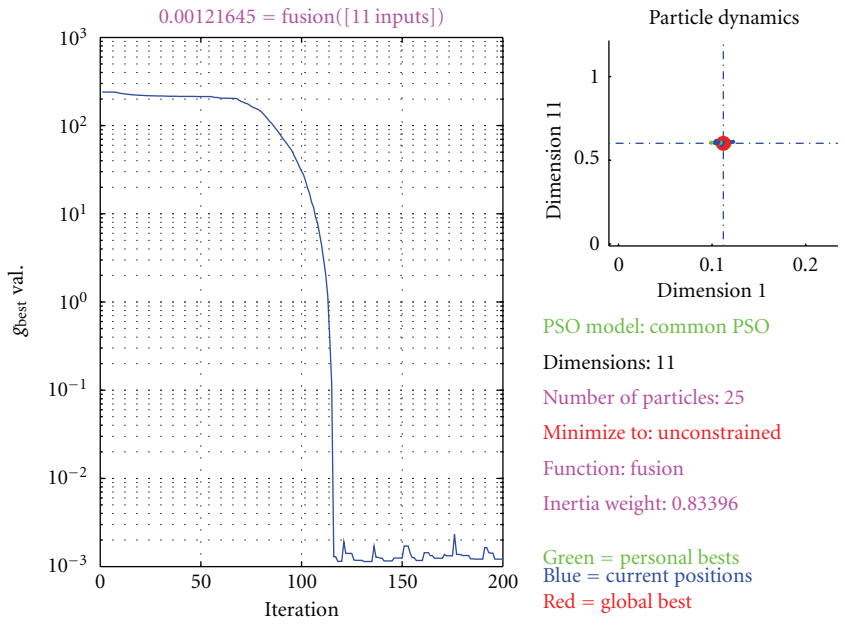
In this section, we would like to demonstrate the performance of our decision fusion rule. On the platform Matlab, the basic PSO algorithm is realized and the objective function is set as $R' = (C_{10}q/C_{01}p)\alpha_g + \beta_g + M(\sum_{i=1}^{n} k_i - 1)^2$. In order to obtain the smallest R', a swarm consists of 25 particles flying around in an $(n+1)$-dimensional search space, where n is the number of network sensors. In the end of searching process, the g_{best} should be the smallest R', and then, with R', the smallest error probability R for our fusion rule can be obtained. Consequently, the corresponding k_i and λ_g are the optimal parameters of the fusion rule.

Firstly, whether the parameters of PSO can affect the performance of our decision fusion rule is demonstrated in Section 4.1. Then, in Section 4.2, let us show that what would happen to our experimental results as the number of sensors increases.

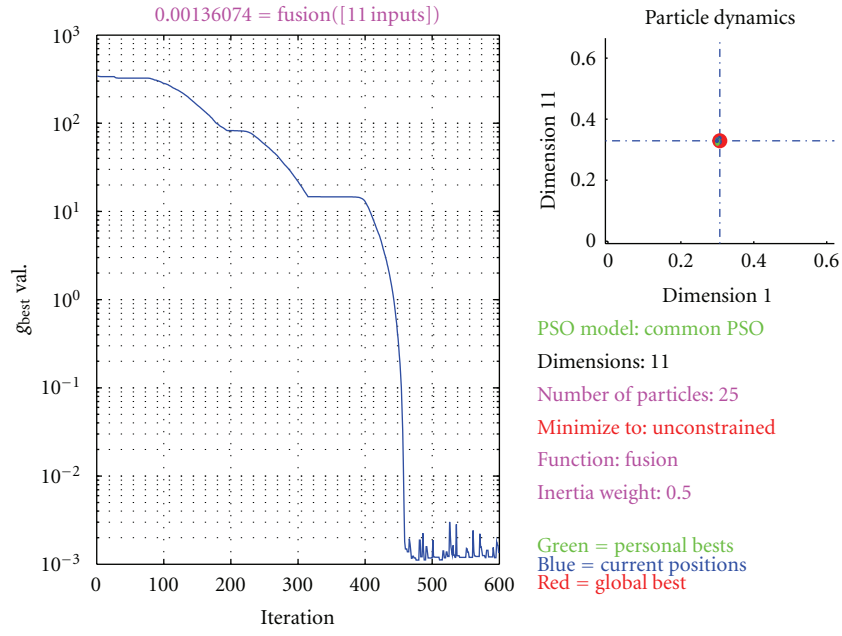
4.1. Choosing PSO Parameters w, c_1, and c_2. It can be seen from expression (1) that the optimal performance of PSO depends on the values of parameters w, c_1, and c_2. The inertia parameter w controls the momentum of each particle. A larger w can increase the global exploration ability while a smaller w tends to improve the local exploration ability of particles. The cognitive learning parameter c_1 limits the contribution of a specific particle p_{best} while the social learning parameter c_2 limits the contribution of the entire swarm g_{best}.

At first, let us discuss the performance of our fusion rule when inertia parameter w is fixed or changes along with time. In the first case, we set $w = 0.5$ while, in the second case, we set $w = w_{\max} - ((w_{\max} - w_{\min})/T_{\max}) \times t$, where t and $T_{\max}$ are, respectively, the current and maximum number of iterations and $w_{\max}$ and $w_{\min}$ are, respectively, the upper and lower bounds of w. In this paper, we set $w_{\max} = 0.9$, $w_{\min} = 0.4$, and $T_{\max} = 1500$. The inputs and outputs of this simulation are as shown in Table 3.

From Table 3 and Figure 3, we can see that when the inertia parameter w changes along with time, the total error



(a) time-variable w



(b) fixed w

Figure 3: The simulation results with time-variable and fixed inertia parameter w.

probability R is lower and the time consumed for convergence is shorter than those when w is fixed. The reason is that if we choose the time-variable inertia parameter, particles can have good global exploration ability at the beginning phase of the searching process and good local exploration ability as the current number of iterations increases. Thereby, we decide to choose the time-variable inertia parameter w.

After that, let us discuss the learning parameters c_1 and c_2. If $c_1 = 0$ or $c_2 = 0$, it means that we do not take the cognitive learning process or the society learning process into account. Under this situation, the simulation results are as shown in Table 4.

From Table 4 and Figure 4, it can be seen that when the cognitive learning parameter c_1 is not taken into account, PSO still can get converged, but the total error probability R becomes larger and time consumed for convergence is longer comparing with the situation when $c_1 \neq 0$. When we do not consider the society learning parameter c_2, PSO cannot get

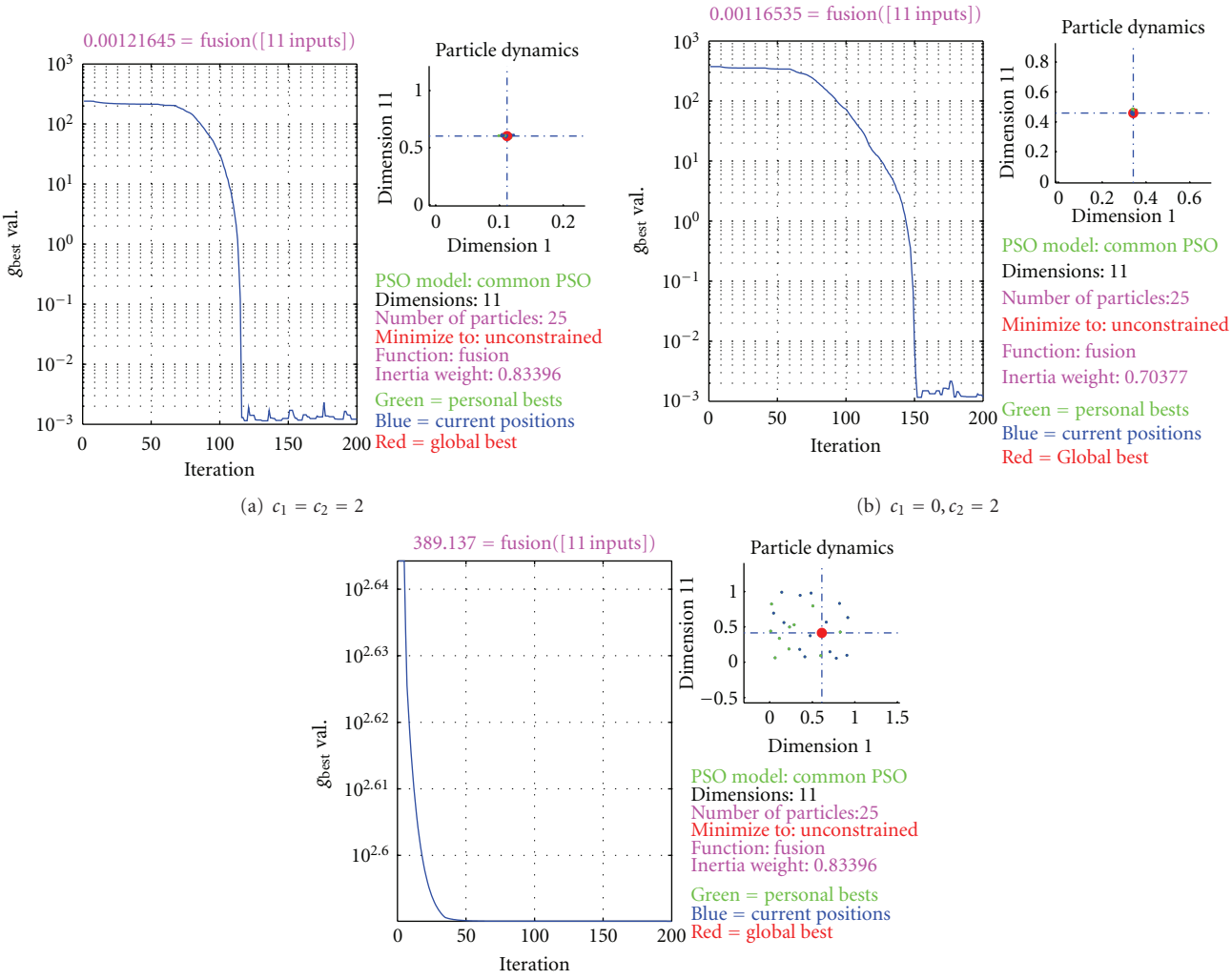


Figure 4: The experimental results with different learning parameters.

converged at all. The reason is that each particle only has the knowledge about itself, so the swarm can never evolve to the same direction.

In sum, from the analysis above, we can see that the time-variable inertia parameter w is better than the fixed one and the situation that both learning parameters c_1 and c_2 are taken into account is better than the situation when $c_1 = 0$ or $c_2 = 0$. Therefore, in the following simulation, the parameters are set as

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T_{\max}} \times t, \qquad c_1 = c_2 = 2. \tag{23}$$

4.2. The Performance of Linear Fusion Rule with Different Sensor Numbers. Here, the number of sensors in the network is set as $n = 5$, $n = 10$ and $n = 15$, respectively. According to the experimental results, it can be seen that our decision fusion rule can indeed get very high accuracy. The inputs and outputs of our simulation are as shown in Table 5 and Figure 5.

When $n = 5$, the decision fusion rule becomes

$$u_g = \begin{cases} 1, & \text{when } 0.102567u_1 + 0.463625u_2 + 0.0806701u_3 \\ & \quad + 0.019778u_4 + 0.333359u_5 > 0.704058, \\ 0, & \text{when } 0.102567u_1 + 0.463625u_2 + 0.0806701u_3 \\ & \quad + 0.019778u_4 + 0.333359u_5 < 0.704058, \end{cases} \tag{24}$$

and, with these parameters, we can get the total error probability $R = 0.0342967$ which is very small. After about more than 50 iterations, the PSO has already got converged, so the time consumed is very short.

Similarly, from Table 5 and Figure 5(b), when $n = 10$, we can see that the total error probability becomes $R = 0.00116502$ and after about 120 iterations the PSO gets

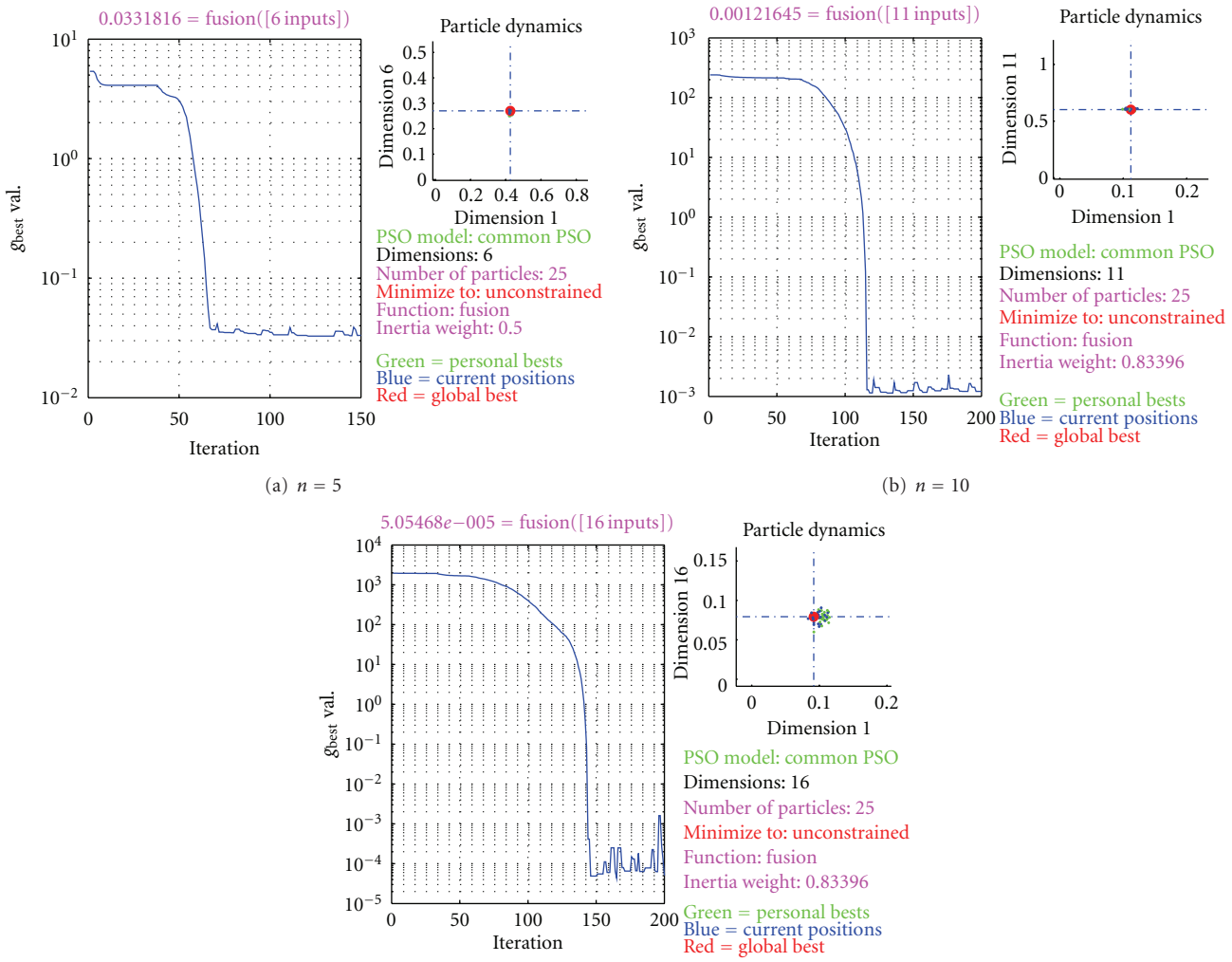


Figure 5: The experimental results under different sensor numbers n.

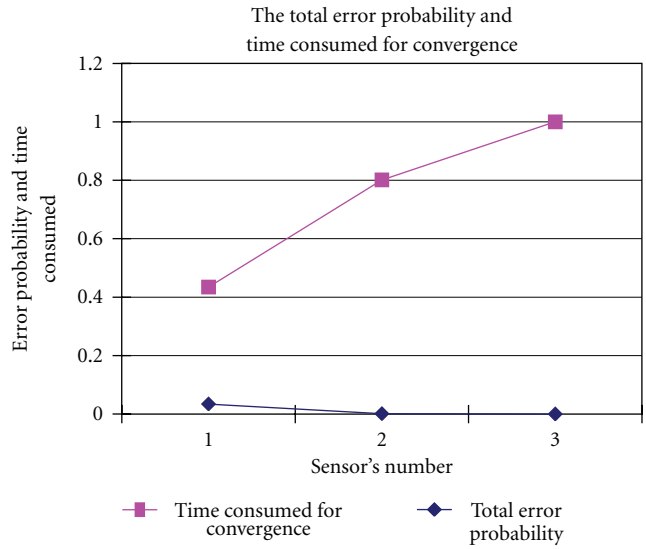


Figure 6: The total error probability R and time consumed for convergence change along with sensor number.

converged. When $n = 15$, $R = 0.0000403$ which is close to zero and after 150 iterations the PSO gets converged.

As shown in Figure 6, along with the increase of n, the error cost becomes smaller and smaller. This phenomenon is easy to understand because the decision result is more precise when there are more sensors working in the network. However, as n increases, the time consumed for PSO to find the optimized parameters becomes longer, so when this fusion rule is used we should better get a balance between accuracy and time.

In brief, this simulation indicates that our linear fusion rule is capable of getting really small total error probability under the control of the constrained PSO, also showing that PSO does very well for the parameter optimization of the linear decision fusion rule.

5. Conclusion

In this paper, we present a linear decision fusion model and propose a way of controlling the parameters of the model taking the advantage of the constrained PSO. In the model, the integrated contribution of local decisions is computed with a linear equation which is made up with local decision weights and local decisions, and then the integrated contribution is compared with a threshold in the fusion center. After that, according to the comparison results, the final decision can be made. Furthermore, the constrained PSO is creatively employed to discover the weights and the threshold. The simulation results show that our linear decision rule and the way of parameter optimization are efficient to get very high accuracy.

References

[1] J.-J. Xiao, S. Cui, Z.-Q. Luo, and A. J. Goldsmith, "Joint estimation in sensor networks under energy constraints," *Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON '04)*, pp. 264–271, October 2004.

[2] L. Snidaro, R. Niu, P. K. Varshney, and G. L. Foresti, "Sensor fusion for video surveillance," in *Proceedings of the 7th International Conference on Information Fusion (FUSION '04)*, pp. 564–571, Stockholm, Sweden, June 2004.

[3] A. Tiwari, F. L. Lewis, and S. S. Ge, "Wireless sensor network for machine condition based maintenance," *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision (ICARCV '04)*, Amari Watergate Bangkok, vol. 1, pp. 461–467, December 2004.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.

[5] N. A. B. A. Aziz, A. W. Mohemmed, and B. S. Daya Sagar, "Particle swarm optimization and Voronoi diagram for wireless sensor networks coverage optimization," in *Proceedings of the International Conference on Intelligent and Advanced Systems (ICIAS '07)*, pp. 961–965, November 2007.

[6] J. Hu, J. Song, M. Zhang, and X. Kang, "Topology optimization for urban traffic sensor network," *Tsinghua Science and Technology*, vol. 13, no. 2, pp. 229–236, 2008.

[7] J. Li, K. Li, and W. Zhu, "Improving sensing coverage of wireless sensor networks by employing mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO' 07)*, pp. 899–903, Sanya, China, December 2007.

[8] X. Wang, S. Wang, and J. J. Ma, "An improved co-evolutionary particle swarm optimization for wireless sensor networks with dynamic deployment," *Sensors*, vol. 7, no. 3, pp. 354–370, 2007.

[9] T. P. Hong and G. N. Shiu, "Allocating multiple base stations under general power consumption by the particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 23–28, April 2007.

[10] A. Gopakumar and L. Jacob, "Localization in wireless sensor networks using particle swarm optimization," in *Proceedings of the IET Conference on Wireless, Mobile and Multimedia Networks*, pp. 227–230, January 2008.

[11] R. V. Kulkarni, G. K. Venayagamoorthy, and M. X. Cheng, "Bio-inspired node localization in wireless sensor networks," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '09)*, pp. 205–210, San Antonio, Tex, USA, October 2009.

[12] K. S. Low, H. A. Nguyen, and H. Guo, "A particle swarm optimization approach for the localization of a wireless sensor network," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE '08)*, pp. 1820–1825, July 2008.

[13] H. Guo, K.-S. Low, and H.-A. Nguyen, "Optimizing the localization of a wireless sensor network in real time based on a low-cost microcontroller," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 3, pp. 741–749, 2011.

[14] K. S. Low, H. A. Nguyen, and H. Guo, "Optimization of sensor node locations in a wireless sensor network," in *Proceedings of the 4th International Conference on Natural Computation (ICNC '08)*, pp. 286–290, October 2008.

[15] S. M. Guru, S. K. Halgamuge, and S. Fernando, "Particle swarm optimisers for cluster formation in wireless sensor networks," in *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, S. K. Halgamuge, Ed., pp. 319–324, December 2005.

[16] N. M. A. Latiff, C. C. Tsimenidis, and B. S. Sharif, "Energy-aware clustering for wireless sensor networks using particle swarm optimization," in *Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, pp. 1–5, September 2007.

[17] X. Cao, H. Zhang, J. Shi, and G. Cui, "Cluster heads election analysis for multi-hop wireless sensor networks based on weighted graph and particle swarm optimization," in *Proceedings of the 4th International Conference on Natural Computation (ICNC '08)*, vol. 7, pp. 599–603, October 2008.

[18] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: a brief survey," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 41, no. 2, pp. 262–267, 2011.

[19] T. Wimalajeewa and S. K. Jayaweera, "Optimal power scheduling for correlated data fusion in wireless sensor networks via constrained PSO," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3608–3618, 2008.

[20] K. Veeramachaneni and L. Osadciw, "Swarm intelligence based optimization and control of decentralized serial sensor networks," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '08)*, pp. 1–8, September 2008.

[21] K. Veeramachaneni and L. A. Osadciw, "Dynamic sensor management using multi objective particle swarm optimizer," in *Multisensor, Multisource Information Fusion: Architectures,*

Algorithms, and Applications, vol. 5434 of *Proceedings of SPIE*, pp. 1–12, Orlando, Fla, USA, April 2004.

[22] C. A. Floudas and P. M. Pardalos, "A collection of test problems for constrained global optimization algorithms," in *Lecture Notes in Computer Science*, vol. 455, p. 180, Springer, Berlin, Germany, 1987.

[23] D. M. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill, New York, NY, USA, 1972.

[24] S. S. Rao, *Optimization: Theory and Applications*, Wiley Eastern Limited, 1977.

[25] J.-M. Yang, Y.-P. Chen, J.-T. Horng, and C.-Y. Kao, "Applying family competition to evolution strategies for constrained optimization," in *Lecture Notes in Computer Science*, vol. 1213, pp. 201–211, Springer, Berlin, Germany, 1997.