In this work, a particle swarm optimization (PSO) algorithm is used to cooperatively estimate a monitored parameter by sensor nodes in an ad-hoc wireless sensor network (WSN). In the proposed algorithm, every sensor node of a wireless sensor network is equipped with a modified particle swarm optimization (MPSO) algorithm to estimate a parameter of interest. A diffusion scheme is used to cooperatively estimate this parameter by sharing the local best particle and the corresponding particle error value to the neighboring nodes. Thus the performance of the wireless sensor network is improved by exploiting the spatial and temporal diversity of the network by collaboratively estimating this parameter. The simulation results show that the diffusion MPSO (DMPSO) algorithm outperforms the non-cooperative MPSO (NCMPSO) algorithm, the diffusion least-mean-squares (DLMS) algorithm and the diffusion recursive-least-squares (DRLS) algorithm by considerable margin.

Index Terms— Wireless sensor network (WSN), particle swarm optimization (PSO), cooperative parameter estimation, diffusion.

1. INTRODUCTION

In recent years, the rapid growth in wireless communication and electronic industry has enabled the development of power-efficient, low-cost and multi-functional wireless sensor networks [1]. This research area has become quite demanding as WSN are being used in numerous applications like environment and habitat monitoring, structural health monitoring, health care, home automation, traffic surveillance, just to name a few. These applications commonly require to estimate certain parameters such as temperature, concentration of chemicals, pressure, speed and position of target object.

The sensor nodes can perform multiple operations like data acquisition from the surrounding physical media, signal processing tasks, control signaling with the central node or with the neighboring nodes and also communicate relevant data collected through wireless transceivers. Usually in a WSN there is a group of sensors nodes in target sensing areas like battle fields, forests etc with limited communication and power capability. In such environments, it becomes difficult to replenish the resources like the battery power of the sensor node, therefore the available resources have to be utilized efficiently. The limited resources can be optimally utilized in a distributed network as it reduces multi hop or long range communication required in the centralized network to send the data to the central nodes for data processing.

Distributed computing has attracted many researchers to apply to WSNs, as it enables low-cost estimation of the parameters and also its robustness to node failure. Distributed parameter estimation can be done either by cooperatively estimating a parameter of interest by data sharing among the neighboring sensor nodes or by non-cooperative estimation where the sensor nodes independently estimates the parameter without any data sharing to its neighboring nodes. An improved performance can be achieved by the collaboration in a cooperative distributed network because it exploits the spatial and temporal diversity of the network to reduce the estimation error whereas the non-cooperative network can only exploit the temporal diversity of the network. There has been extensive research done and different data sharing schemes proposed to do cooperative parameter estimation to exploit the distributed nature of the network effectively.

Initially, an incremental scheme for distributed data processing was suggested [2], wherein the information circulated through a topological cycle and the least-mean-square (LMS) algorithm [3] was used at each sensor to adapt to variations in the signal statistics. This distributed scheme provides a faster convergence than a centralized scheme as well as a lower steady-state error at a lower computational complexity. However, if any sensor node in the sequential cycle fails then the network stops functioning until the cycle is restored. In [4], a solution was suggested to the node failure problem at the cost of higher computational complexity.

To overcome the drawbacks of the incremental algorithm and fully exploit the distributive nature of the network, a diffusion-based LMS algorithm was proposed in [5]. In the diffusion LMS (DLMS) algorithm, sensors share data with their neighbors and perform local estimation using shared data with their nearby, single-hop neighbors and perform local estimation using the shared information. The DLMS algorithm is computationally more complex than the incremental
algorithm and is inferior in performance as well. However, it is robust to node failure, which makes it more suitable for use in practical applications.

Particle swarm optimization, a modern heuristic algorithm belongs to the category of swarm intelligence methods, was first introduced by Kennedy and Eberhart [6]. The PSO algorithm has advantages such as ease of hardware and software implementation, available guidelines for choosing its parameters, ability to overcome local minimum problem and faster convergence than other heuristic algorithms such as genetic algorithm, differential evolution and bacterial foraging algorithm [8]. In this work, a standard PSO algorithm [7] is modified by using the inertia weight update function proposed in [9] to increase the convergence speed of the particles in the search space. This modified PSO algorithm is used at every node to estimate the parameter of interest. In the earlier optimization problems related to WSNs [10], the PSO algorithm used a larger particle swarm size and the data window size to estimate the parameter at every node. Whereas in the proposed algorithm, the particle swarm size and data window size are reduced by cooperatively estimating the parameter using a novel diffusion scheme and thus the computational complexity is reduced.

The paper is organized as follows. The problem is formulated in Section 2. followed by the network model discussed in Section 3. The proposed algorithm is detailed in Section 4. In Section 5, the simulation results are reported and the paper concludes with Section 6.

2. PROBLEM FORMULATION

Consider a group of \( S \) sensor nodes randomly distributed in the target sensing area. At the sensor node an unknown system parameter \( w_0 \) is estimated. The unknown system parameter \( w_0 \) is represented by a column vector of order \( m \times 1 \). The input and output of the system at time \( t \) is defined by \( U_s(t) \) and \( d_s(t) \), respectively. The input data matrix \( U_s(t) \) is of order \( n \times m \) and is a group of row vectors \( u_s(t) \) of order \( m \times 1 \) formed using the input data block as follows:

\[
U_s(t) = \text{col}\{u_s(t-n+1), u_s(t-n+2), \ldots, u_s(t)\}
\]

and \( d_s(t) \) is a column vector of length \( n \times 1 \) and is expressed as follows:

\[
d_s(t) = U_s(t)w_0 + v_s(t),
\]

where \( v_s(t) \) is additive noise. Generally a PSO algorithm works efficiently for batch type optimization problems where the entire input data is available off line. But in an online processing scenario the entire input data is not available, so an input data block of size \( n \) is taken at every iteration. The data window slides at every iteration by one step which will add a new data point and exclude the oldest data point in the data window so that the window size remains constant.

Here, in this work a MPSO algorithm is used at each node to minimize the objective function defined as

\[
J_{s,i} = \frac{||d_s - U_s X_{s,i}||^2}{n},
\]

where \( X_{s,i} \) is the \( i^{th} \) particle position vector of node \( s \). This objective function defines the search space and the position of every particle in the search space is assumed to be the potential estimate of the vector \( w_0 \). In the proposed diffusion scheme the sensor nodes share its local best particle position \( X_{s,i}^* \) and the corresponding local best error \( J_{s,i}^* \) with its neighboring nodes as shown in Fig. 1. The information shared from the neighboring nodes is used to reduce the estimation error at every node.

![Fig. 1. Neighboring nodes sharing their local best particle](image)

3. NETWORK MODEL

Consider a network of \( S \) sensor nodes randomly distributed in a normalized area. The nodes are placed in such a way that every node has some sensors in close proximity and each node is interconnected only to its neighboring nodes. Each node forms a communication link with its neighbors to share information in a single hop. The communication range \( r \) is set based on the amount of transmitting power each node is allowed. So the nodes that are within the range \( r \) of any node \( s \) comprise the neighbors of that node. It is also assumed that the communication between nodes is noise free.

4. THE DIFFUSION MODIFIED PSO ALGORITHM

In the diffusion modified PSO algorithm, a modified PSO algorithm is used at every node \( s \) to estimate the desired parameter \( w_0 \) and a novel diffusion scheme is used to do the estimation cooperatively which improves the network performance. The steps of the proposed algorithm are as follows:

1. **Initialization**: At \( t = 0 \), initialize \( k \) particles \( X_{s,i}(0), i = 1, 2, \ldots, k \) of dimension \( m \) at each node, where \( X_{s,i}(0) = \)
Particle error calculation:

Particle best position:

Diffusion:

The coefficients \( x_{s,i,j}(0) \), \( j = 1, 2, \ldots, m \) of every particle are uniformly distributed in the range \([x_{\text{min}}, x_{\text{max}}]\). Similarly, initialize the velocities \( v_{s,i,j}(0) \), \( i = 1, 2, \ldots, k \) of all the node particles, where \( v_{s,i,j}(0) = [v_{s,i,j}(1), v_{s,i,j}(2), \ldots, v_{s,i,j}(m)] \). The velocity coefficients \( v_{s,i,j}(0) \) are uniformly distributed in the range \([-v_{\text{max}}, v_{\text{max}}] \). The velocity coefficients are limited in a certain range to explore the search space more effectively and the maximum velocity coefficient \( v_{\text{max}} \) is defined as [11]

\[
v_{\text{max}} = v_c \cdot x_{\text{max}},
\]

where \( v_c \) is the velocity constraint factor.

2. Particle error calculation: Calculate the estimation error for every particle using the objective function given in (3).

3. Particle best position: Only in first iteration \((t = 0)\), set the particle best position \( X_{s,i}^*(0) \) to the current position of the particle \( X_{s,i}(0) \) and particle best error \( J_{s,i}^*(0) \) to the corresponding particle error value \( J_{s,i}(0) \). For \( t > 0 \), check: If \( J_{s,i}(t) < J_{s,i}^*(t-1), i = 1, 2, \ldots, k \), then set \( J_{s,i}^*(t) = J_{s,i}(t), X_{s,i}^*(t) = X_{s,i}(t) \) and continue; else set \( J_{s,i}^*(t) = J_{s,i}(t-1), X_{s,i}^*(t) = X_{s,i}(t-1) \) and continue.

4. Local best particle position: Search for the minimum among all particle best error \( J_{s,i}^*(t), i = 1, 2, \ldots, k \) and assign it to \( J_\text{s, min}(t) \), then set \( X_\text{s, min}(t) \) to the particle position corresponding to the error \( J_\text{s, min}(t) \). If \( t > 0 \) and \( J_\text{s, min}(t) < J_{s,i}^*(t-1) \), then update local best particle error as \( J_{s,i}^*(t) = J_\text{s, min}(t) \) and local best particle position as \( X_{s,i}^*(t) = X_\text{s, min}(t) \) and continue; else set \( J_{s,i}^*(t) = J_{s,i}^*(t-1), X_{s,i}^*(t) = X_{s,i}^*(t-1) \) and continue.

5. Diffusion: If a node has \( p \) neighboring nodes including itself then share its local best particle error \( J_{s,i}^* \) and corresponding local best particle position \( X_{s,i}^* \) to its \( p - 1 \) neighboring nodes. Using the error values received from its neighbors, as shown in Fig. 1, identify the minimum local best error among itself and \( p - 1 \) neighboring nodes and set \( J_{s,i}^*(t) = \min(J_{s,i}^*(t), J_1^*(t), \ldots, J_{p-1}^*(t)) \) and then update the local best particle position \( X_{s,i}^*(t) \) to the particle position corresponding to the error \( J_{s,i}^*(t) \).

6. Velocity update: For the next iteration update the particle velocity using the current particle velocity, the local best particle position \( X_{s,i}^* \) and particle best position \( X_{s,i}^* \). The \( i^{th} \) particle velocity coefficient in the \( j^{th} \) di-

mension is updated according to

\[
v_{s,i,j}(t) = \begin{cases} \frac{J_{s,i,j}(t) \cdot J_{s,i,j}(t-1) + c_1 \cdot J_{s,i,j}(t-1) - x_{s,i,j}(t-1)}{c_1 \cdot \Delta J_{s,i,j}(t)}, & \text{if } J_{s,i,j}(t) < \frac{J_{s,i,j}(t-1) - x_{s,i,j}(t-1)}{\Delta J_{s,i,j}(t)} \\ c_2 \cdot \Delta J_{s,i,j}(t) - x_{s,i,j}(t-1), & \text{otherwise} \end{cases}
\]
modification however, does not prevent the hill climbing capabilities of PSO, it merely increases the influence of potentially fruitful inertia directions, while decreasing the influence of potentially unfavorable inertia directions.

Goto step 2.

Repeat steps 2 to 10 at every node \( s, s = 1, 2, \ldots, S \). Figure 2 details the steps of the DMPSO algorithm.

![Fig. 2. DMPSO algorithm execution steps.](image)

5. SIMULATION RESULTS

The proposed algorithm is simulated with a network size of \( S = 20 \) sensor nodes. The input data is correlated with unit variance and the noise is assumed to be additive white Gaussian with zero mean and unit variance. The unknown vector \( \mathbf{w}_0 \) \( m \times 1 \) is initialized as \( \text{col} \{ 1, 1, \ldots, 1 \} / \sqrt{m} \), input data block \( n = 10 \) and the tap size \( m = 4 \). At every node a swarm size of \( k = 5 \) particles is initialized. The swarm parameters such as the acceleration constants \( c_1 \) and \( c_2 \) are set to 1.8, the inertia weight \( \text{inertia}_w \) is 1 and the slope constant \( S_j = 1.2 \). The mean-square-deviation (MSD) is calculated at every node using the local best particle position \( X_{s, k}^* \) according to:

\[
\text{MSD} = E[|w_0 - X_{s, k}^*|^2].
\]

Figure 3 depicts the performance of the DMPSO algorithm as compared to that of the non-cooperative MPSO algorithm (NCMPSO). As can be seen from this figure, a 15 dB improvement is brought about by the DMPSO algorithm over the NCMPSO algorithm at both 10 dB and 20 dB SNRs. The poor performance of the NCMPSO algorithm is due to its convergence to a local minimum. The cooperative estimation improves the performance as the proposed algorithm exploits both the spatial and temporal diversity of the network.

The DMPSO algorithm also reduces the computational complexity and improves the performance of PSO algorithm by using smaller swarm size at sensor nodes. This is demonstrated by a scenario where 100 particles at each node is used in a non-cooperative sensor network and 5 particles at each node is used in a cooperative sensor network. For the non-cooperative network the MPSO algorithm is used and for cooperative network DMPSO algorithm is used. The results are reported in Fig. 4. This figure shows that by doing cooperative estimation the computational complexity can be reduced without any performance degradation of the network.

Finally, the performance of the DMPSO algorithm is compared to those of the DLMS algorithm [5] and the DRLS algorithm [12]. The results are depicted in Fig. 5 and Fig. 6 for SNR of 10 dB and 20 dB, respectively. As can be seen from these figures, the DMPSO is by far outperforming both of these algorithms. This improvement is of course reached at an extra computational complexity.

6. CONCLUSION

This work proposes a new algorithm for parameter estimation in adaptive wireless networks. The well-documented diffusion scheme is combined with a modified PSO algorithm. The resulting DMPSO algorithm has been shown to outperform both the DLMS and DRLS algorithms.

7. ACKNOWLEDGMENT

The authors would like to acknowledge the support provided by the Deanship of Scientific Research (DSR) at King Fahd University of Petroleum & Minerals (KFUPM) for funding this work through project No. RG1216-1 & RG1216-2.

![Fig. 3. Performance of NCMPSO and DMPSO algorithms.](image)
Fig. 4. Performance of NCMPSO and DMPSO algorithms using different population sizes.

Fig. 5. Performance of DLMS, DRLS and DMPSO algorithms at SNR = 10 dB.

Fig. 6. Performance of DLMS, DRLS and DMPSO algorithms at SNR = 20 dB.

8. REFERENCES


