COOPERATIVE PARAMETER ESTIMATION USING PSO IN AD-HOC WSN

Sameer Husain Arastu, Azzedine Zerguine, Muhammad Omer Bin Saeed, and Ali T. Al-Awami

Electrical Engineering Department, King Fahd University of Petroleum & Minerals
Dhahran 31261, Saudi Arabia
Email: {sameerh, azzedine, mobs, aliawami}@kfupm.edu.sa

ABSTRACT

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 low 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 $\mathbf{w}_0$ is estimated. The unknown system parameter $\mathbf{w}_0$ is represented by a column vector of order $m \times 1$. The input and output of the system at time t is defined by $\mathbf{U}_s(t)$ and $\mathbf{d}_s(t)$, respectively. The input data matrix $\mathbf{U}_s(t)$ is of order $n \times m$ and is a group of row vectors $\mathbf{u}_s(t)$ of order $m \times 1$ formed using the input data block as follows:

$$\mathbf{U}_s(t) = \text{col}\{\mathbf{u}_s(t-n+1), \mathbf{u}_s(t-n+2), \ldots, \mathbf{u}_s(t)\} \quad (1)$$

and $\mathbf{d}_s(t)$ is a column vector of length $n \times 1$ and is expressed as follows:

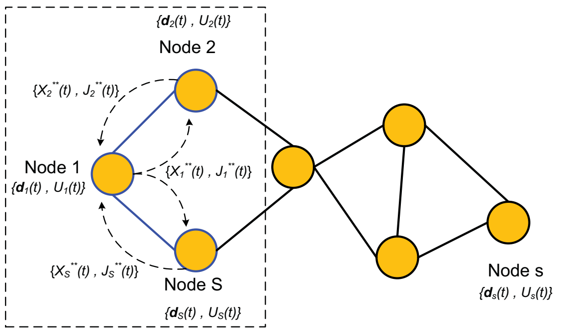
$$\mathbf{d}_s(t) = \mathbf{U}_s(t)\mathbf{w}_0 + \mathbf{v_s}(t), \quad (2)$$

where $\mathbf{v}_s(t)$ 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

$$J_{s,i} = [||\mathbf{d}_s - \mathbf{U}_s X_{s,i}||^2]/n, \quad (3)$$

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 $\mathbf{w}_0$. In the proposed diffusion scheme the sensor nodes share its local best particle position X_s^{**} and the corresponding local best error J_s^{**} 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.



$\{\mathbf{d}_2(t), U_2(t)\}$
Node 2
$\{X_2^{**}(t), J_2^{**}(t)\}$
Node 1
$\{\mathbf{d}_1(t), U_1(t)\}$
$\{X_1^{**}(t), J_1^{**}(t)\}$
$\{X_S^{**}(t), J_S^{**}(t)\}$
Node S
$\{\mathbf{d}_S(t), U_S(t)\}$
Node s
$\{\mathbf{d}_s(t), U_s(t)\}$

Fig. 1. Neighboring nodes sharing their local best particle X_s^{**} and the corresponding error J_s^{**}.

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 $\mathbf{w}_0$ 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 $X_{s,i}(0), i = 1, 2, ..., k$ of dimension m at each node, where $X_{s,i}(0) =$

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

$$v_{\max} = v_c \cdot x_{\max}, \tag{4}$$

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 $J_{s,i}(t) < J^*_{s,i}(t-1), i = 1, 2, ..., k$ 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 $J^*_{s,i}(t), i = 1, 2, ..., k$ and assign it to $J_{s,\min}(t)$, then set $X_{s,\min}(t)$ to the particle position corresponding to the error $J_{s,\min}(t)$. If $t > 0$ and $J_{s,\min}(t) < J^{**}_s(t-1)$, then update local best particle error as $J^{**}_s(t) = J_{s,\min}(t)$ and local best particle position as $X^{**}_s(t) = X_{s,\min}(t)$ and continue; else set $J^{**}_s(t) = J^{**}_s(t-1)$, $X^{**}_s(t) = X^{**}_s(t-1)$ and continue.

5. **Diffusion:** If a node has p neighboring nodes including itself then share its local best particle error J^{**}_s and corresponding local best particle position X^{**}_s 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 $J^{**}_s(t) = \min(J^{**}_s(t), J^{**}_1(t), ..., J^{**}_{p-1}(t))$ and then update the local best particle position X^{**}_s to the particle position corresponding to the error $J^{**}_s(t)$.

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} dimension is updated according to

$$\begin{aligned} v_{s,i,j}(t) &= iw_{s,i}(t)\, v_{s,i,j}(t-1) \\ &\quad + c_1 r_1 \left(x^*_{s,i,j}(t-1) - x_{s,i,j}(t-1)\right) \\ &\quad + c_2 r_2 \left(x^{**}_{s,j}(t-1) - x_{s,i,j}(t-1)\right) \end{aligned} \tag{5}$$

where c_1 and c_2 are acceleration constants and r_1 and r_2 are uniformly distributed random numbers in $[0, 1]$. In (5), the first term represents the inertia part, which controls the global and local exploration of the particles in the search space, the second term represents the cognitive part of PSO where the particle changes its velocity based on its own thinking and memory. The third term represents the social part of the PSO where the particle changes its velocity based on the social-psychological adaption of knowledge [7]. The acceleration constants are used to control the speed of advancement of the particles towards the particle best and local best position and thus prevent the particles from stagnation. For all particles limit the velocity coefficients to the predefined velocity limit $v_{\max}$.

7. **Position update:** Using the updated velocities, then update the particle position according to:

$$x_{s,i,j}(t) = x_{s,i,j}(t-1) + v_{s,i,j}(t). \tag{6}$$

8. **Stopping criteria:** If the maximum number of allowable iterations is reached then stop; else continue.

9. **Time update:** Update the time counter $t = t + 1$.

10. **inertia weight update:** Update the inertia weight according to [9]:

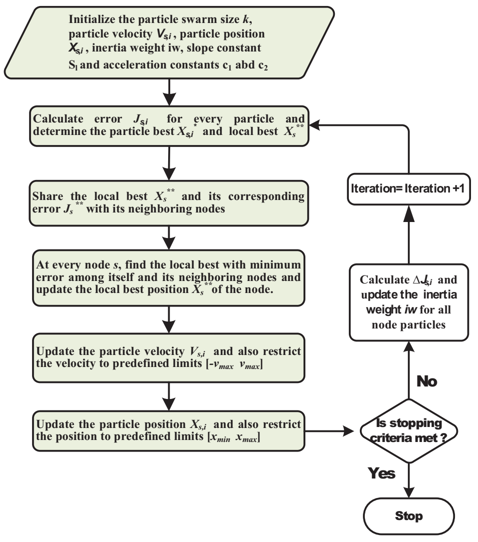
$$iw_{s,i}(t) = \frac{1}{\left(1 + e^{\frac{-\Delta J_{s,i}(t)}{S_l}}\right)}, \tag{7}$$

where $iw_{s,i}(t)$ is the inertia weight of the i^{th} particle of node s, $\Delta J_{s,i}(t)$ is the change in particle error between the current and last generation, and S_l is the slope constant used to adjust the transition slope based on the expected error range. This relation limits the inertia weight in the interval (0,1), with the midpoint of 0.5 corresponding to zero change in error. Consequently, increase in error will lead to inertia weight larger than the recommended fixed experimental value of 0.5, and decrease in error will lead to inertia weight smaller than 0.5.

The inertia weight is made adaptable i.e. it is either maintained at same value or changed when a better particle position is encountered to move the particle more closer to the favorable position. If the particle does not attain a lower error, its inertia influence is reduced. This

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 $s, s = 1, 2, ..., S$. Figure 2 details the steps of the DMPSO algorithm.



Initialize the particle swarm size k, particle velocity $V_{s,i}$, particle position $X_{s,i}$, inertia weight iw, slope constant S_l and acceleration constants c_1 abd c_2
Calculate error $J_{s,i}$ for every particle and determine the particle best $X_{s,i}^*$ and local best X_s^{**}
Share the local best X_s^{**} and its corresponding error J_s^{**} with its neighboring nodes
At every node s, find the local best with minimum error among itself and its neighboring nodes and update the local best position X_s^{**} of the node.
Update the particle velocity $V_{s,i}$ and also restrict the velocity to predefined limits [$-v_{max}$ v_{max}]
Update the particle position $X_{s,i}$ and also restrict the position to predefined limits [x_{min} x_{max}]
Is stopping criteria met ?
No
Calculate $\Delta J_{s,i}$ and update the inertia weight iw for all node particles
Iteration= Iteration +1
Yes
Stop

Fig. 2. DMPSO algorithm execution steps.

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 $col\{1, 1, ..., 1\}/\sqrt{m}$, 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 $iw_{s,i}$ = 1 and the slope constant S_l = 1.2. The mean-square-deviation (MSD) is calculated at every node using the local best particle position X_s^{**} according to:

$$\text{MSD} = \mathsf{E}[||\mathbf{w_0} - X_s^{**}||^2]. \qquad (8)$$

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-documenteted 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**.

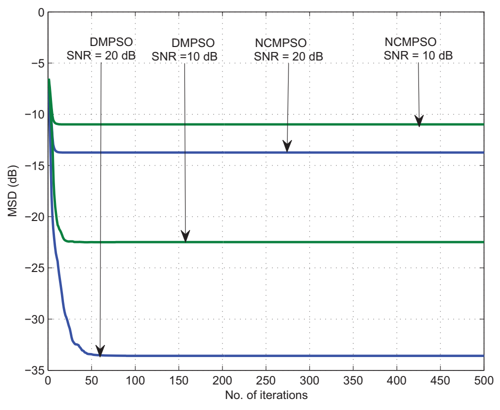


Fig. 3. Performance of NCMPSO and DMPSO algorithms.

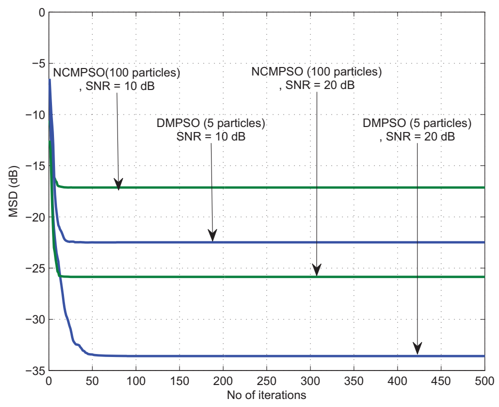


Fig. 4. Performance of NCMPSO and DMPSO algorithms using different population sizes.

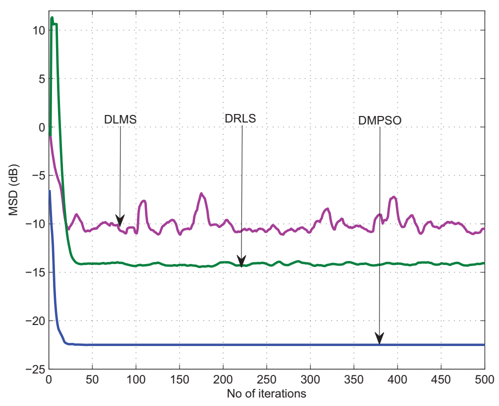


Fig. 5. Performance of DLMS, DRLS and DMPSO algorithms at SNR = 10 dB.

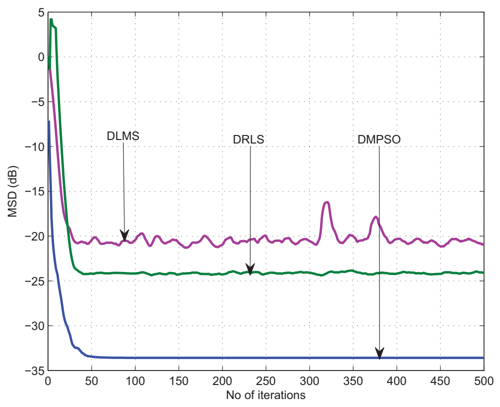


Fig. 6. Performance of DLMS, DRLS and DMPSO algorithms at SNR = 20 dB.

8. REFERENCES

[1] I. F. Akyildiz, "A survey on sensor networks," IEEE Comm. Mag., vol. 40, no. 8, pp. 102-114, 2002.

[2] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," IEEE Trans. on Signal Processing, vol. 55, pp. 4064-4077, Aug. 2007.

[3] S. Haykin, Adaptive Filter Theory, 3rd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1996.

[4] C. G. Lopes and A. H. Sayed, "Randomized incremental protocols over adaptive networks," IEEE ICASSP, pp. 3514-3517, 2010.

[5] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: formulation and performance analysis," IEEE Trans. on Signal Process., vol. 56, no. 7, pp. 3122-3136, July 2008.

[6] J. Kennedy and R. Eberhart, Swarm intelligence, Academic Press, 2001.

[7] Y. Shi, R. Eberhart, "A modified particle swarm optimizer," IEEE Int'l Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence., vol., no., pp.69-73, 4-9 May 1998

[8] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," IEEE Trans. on Evolutionary Computing, vol. 8, no. 3, pp. 211-224, Aug. 2004.

[9] D. J. Krusienski and W. K. Jenkins, "A modified particle swarm optimization algorithm for adaptive filtering," IEEE Int'l Symposium on Circuits and Systems, vol., no., pp.4 pp.-140, 21-24 May 2006.

[10] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle Swarm Optimization in Wireless-Sensor Networks: A Brief Survey," IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol.41, no.2, pp.262-267, March 2011.

[11] Y. Shi, R. C. Eberhart, "Comparing inertia weights and constriction factors in particle swarm optimization," Proceedings of the 2000 Congress on Evolutionary Computation., vol.1, no., pp.84-88 vol.1, 2000

[12] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion Recursive least squares for Distributed Estimation Over Adaptive Networks," IEEE Trans. on Signal Process., vol. 56, no. 5, pp. 1865-1877, May 2008.