IJICTR International Journal of Information & Communication Technology Research

Volume 4- Number 2- March 2012 (1- 10)

Optimized Regression in Sensor Networks by Integration of Harmony Search and Particle Swarm Optimization

Hadi Shakibian
Fac. of Electrical and Computer Engineering
Tarbiat Modares University
Tehran, Iran
h.shakibian@modares.ac.ir

Nasrollah Moghadam Charkari
Fac. of Electrical and Computer Engineering
Tarbiat Modares University
Tehran, Iran
moghadam@modares.ac.ir

Received: October 2, 2011- Accepted: December 29, 2011

***Abstract*—Regression modeling in sensor networks is a difficult task due to (*i*) the network data is distributed among the nodes and (*ii*) the restricted capabilities of the sensor nodes, such as limited power supply and bandwidth capacity. Recently, some distributed approaches have been proposed based on gradient descent and Nelder-Mead simplex methods. Although in these methods, the energy consumption is low, but the accuracy is still far from the centralized approach. Also, they suffer from a high latency. In this paper, a two-fold distributed approach has been proposed for doing regression analysis in wireless sensor networks. After clustering the network, the regressor of each cluster is learned by the integration of particle swarm optimization and harmony search. Afterwards, cluster heads collaborate to construct the global network regressor using a weighted averaging combination rule. The experimental results show the proposed approach improves the accuracy and latency significantly while its energy consumption is considerably acceptable in comparison with its popular counterparts.**

Keywords-sensor networks; distributed regression; particle swarm optimization; harmony search; multiple classifier systems.

I. INTRODUCTION

Wireless sensor networks consist of many sensor nodes which are spatially distributed in an area. Remote monitoring is the main task of a sensor network in which each sensor node is required to capture some phenomena of interest. Instead of transmitting raw measurements, the sensor nodes can use their own capabilities to carry out local computations and only transmit useful information. In-network information processing can decrease the amount of data transmissions and substantially prolongs the network lifetime. This is because the communication has been found to be the most important factor in consuming the power of each sensor node rather than data processing.

Increasing the amount of the collected data in the network requires developing some efficient techniques in order to extract the useful information. Gathering all the collected data in a fusion center, known as the centralized approach, needs a huge data transmission which causes remarkable decreasing in the network lifetime. For this reason, distributed data analysis has been considered as one of the interesting, and of course, a challenging task in the sensor networks. Particularly, regression modeling has been addressed in this paper. Like other well-known machine learning approaches, regression methods [1] work basically in a centralized environment where both data and

processing are centrally available. In a wireless sensor network, data is distributed among the nodes as well as the processing resources. Hence, employing common regression techniques in such an environment is not straightforward. Besides that, the limitations of the sensor nodes, such as restricted power supply and bandwidth capacity [2], are accomplished the difficulty of doing regression over sensor networks.

Recently, some efforts have been done for regression analysis in sensor networks which try to learn the network model using an indirect optimization based strategy [3, 4], in which a learning problem is considered as an optimization one. The idea of these approaches is to learn the network model incrementally through a pre-established Hamiltonian path among the nodes. An incremental version of the gradient descent optimization technique, denoted as IG, has been proposed by [5]. In IG, firstly, a Hamiltonian path is established among the nodes and then, one iteration of the gradient descent is mapped into a network cycle. An accurate final regressor might be achieved after passing many network cycles. In [6], IG has been improved in terms of the accuracy and latency, by clustering the network and employing incremental gradient within the clusters. By setting up the Hamiltonian path among the cluster heads, the convergence rate is increased [7]. An incremental optimization approach based on Nelder-Mead Simplex method, denoted as IS, has been proposed in [8] and [9] which has been integrated with boosting and re-sampling techniques, respectively. Their results show better accuracy and convergence rate compared to the gradient-based counterparts. However, the accuracy of all distributed approaches is still far from the centralized approach. Also, both of the gradient and Nelder-Mead Simplex based approaches suffer from a high latency due to traversing the network, node by node.

So, improving the accuracy and latency are our main motivation to move to a new distributed approach for doing regression analysis over sensor networks. In this paper, we have proposed a distributed evolutionary based approach which integrates two efficient optimization methods, particle swarm optimization (PSO) and harmony search (HS), to learn the network regressor.

The idea of the proposed approach, HDP henceforth (Harmony Search-Distributed PSO), is to learn the network model in two folds. This approach is an extension of our recently published paper in [10]. Firstly, the network is partitioned into several clusters and the regressor of each cluster is learned, separately. To learn a cluster regressor, a swarm of particles is dedicated to the cluster, and then distributed among the cluster nodes to form several sub-swarms, one per cluster node. The cluster swarm is then optimized through optimization of the sub-swarms using the PSO algorithm. The main challenge in this scheme is to guarantee the convergence of the sub-swarms to the corresponding cluster regressor. In order to deviate with this problem, the HS algorithm has been employed in the cluster head and the final cluster regressor is obtained after the completion of improvisation process. Secondly, the cluster heads collaborate to construct the global model, incrementally. We have compared the proposed approach, with its popular counterparts regarding to the accuracy, latency, and energy efficiency.

Section 2 provides a brief background on the regression analysis, particle swarm optimization, and harmony search. Section 3 describes our assumptions and formulates distributed regression problem in sensor networks. The proposed approach is introduced in section 4. The evaluation and experimental results are discussed in section 5 and the last section is concluding remarks.

II. PRELIMINARIES

A. Regression Modeling

The task of supervised learning is to extract a function from a training data set $DS = \{(x_1, y_1), ..., (x_N, y_N)\}$. Every data point consists of some independent input variables (or features) and the desired dependent output variable (or label). If the output takes its value from a discrete space, the learning problem is classification; otherwise it is called regression. In regression, it is assumed that a model defined up to a set of parameters x as:

$$y = g(x \mid \theta) \tag{1}$$

where $g(.)$ is the model, y is a real-valued label, and θ is a parameter [11]. If the form of $g(.)$ is known, the regression is called parametric in which the learning program has to optimize θ, such that the approximation error be minimized:

$$\theta^* = \arg\min_{\theta} \{ \sqrt{\frac{1}{N} \sum_{i=1}^{N} [g(x_i \mid \theta) - y_i)]^2} \} \tag{2}$$

where θ^* is the optimal parameter of the model, (x_i, y_i) is i-th data point, and N is the size of the training data set. In non-parametric regression, there is not any pre-specified model and the predictions are conducted based on similarities among the data points [11]. In this paper we focus on doing distributed parametric regression in wireless sensor networks.

B. Particle Swarm Optimization (PSO)

There are some optimization techniques which inspired from nature. The PSO algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock [12]. The movement of particles in search space is based on their associated velocity vectors at every time step z:

$$\begin{aligned} v_{ij}(z+1) = {} & w \times v_{ij}(z) \\ & + c_1 \times r_1(z)[pbest_{ij}(z) - p_{ij}(z)] \\ & + c_2 \times r_2(z)[gbest_j(z) - p_{ij}(z)] \end{aligned} \tag{3}$$

where p_{ij}, $pbest_{ij}$, and $gbest_j$ are the position of the particle i, the best position founded by the particle i, and the best particle encountered in the swarm, in dimension j, respectively. The parameter w, which is called as inertia weight, stands for exploration-exploitation trade-off. The parameters c_1 and c_2 are two positive acceleration constants to scale cognitive and social components, respectively, and $r_1(z)$ and $r_2(z)$ are two uniform random numbers at time step z.

Every particle can move to a new position by adding up its new velocity vector to its current position:

$$p_{ij}(z+1) = p_{ij}(z) + v_{ij}(z+1) \quad (4)$$

The equations (3) and (4) are updated iteratively by the particles until the stopping condition is met. The PSO algorithm has a superior performance for exploring real-valued search spaces, e.g. regression analysis [15, 16]. It has been widely used in many optimization problems in wireless sensor networks (e.g. [13, 14]). However, up to our knowledge, it has not been used for learning the network data model.

C. *Harmony Search (HS)*

Harmony Search conceptualizes a behavioral phenomenon of music players in improvisation process [17]. The HS algorithm is a population based optimization technique. In HS, the individuals are denoted by harmonies which are located in Harmony Memory (HM). After HM initialization, new harmonies could be generated based on either considering the HM or permissible range of each decision variable. The steps of HS algorithm are as following [18]:

1. Initialize the optimization problem and algorithm parameters.
2. Initialize the harmony memory.
3. Improvise a new harmony from HM.
4. Update HM.
5. Repeat the steps 3 and 4 until the termination criterion is satisfied.

Generating a new harmony is called improvisation. In step 3, to improvise a new harmony vector with d decision variables, $x' = (x'_1, x'_2, ..., x'_d)$, some probabilistic rules should be considered as depicted in Figure 1. The HS algorithm presents some advantages over the other optimization techniques [19] such as:

- HS imposes a fewer mathematical requirements.
- HS can handle discrete variables as well as continuous ones.
- HS generates a new harmony after considering all of the existing harmonies in HM.

The HS has successfully been used in many optimization problems, and some of its variants have also been introduced [18, 19], recently.

III. Assumptions And Problem Statement

A. *Assumptions*

We consider a sensor network with n sensor nodes which are spatially distributed in a bi-dimensional area. The sensor nodes can localize themselves by performing a localization algorithm [20]. Also, suppose the network has been partitioned into C clusters, designating a cluster head for each one, CH_c (c=1,…,C). Since clustering is not the subject of this paper, we assume the network has been clustered via a well-known clustering algorithm [21]. The communication model of the network has been depicted in Figure 2.

B. *Distributed Regression in Sensor Networks*

Suppose the nodes capture some spatiotemporal measurements. Every sensor node i stores each measurement as a quadruple $<x_i, y_i, t_{i,j}, l_{i,j}>$, where (x_i,y_i) is the location of node i, and $l_{i,j}$ is the captured phenomenon of interest (e.g. temperature) by node i at

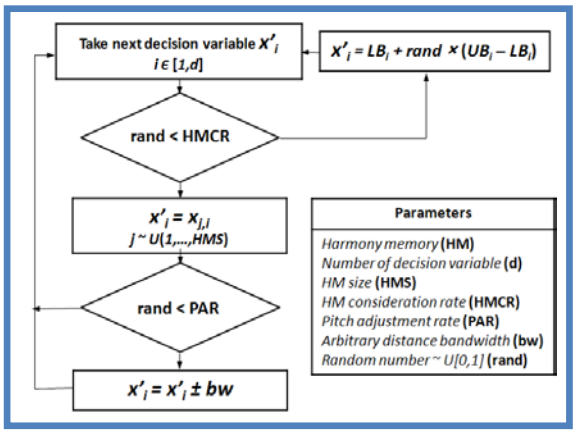

Figure 1. A new harmony is improvised using three probabilistic rules based on *HMCR*, *PAR*, and (1-*HMCR*) probabilities.

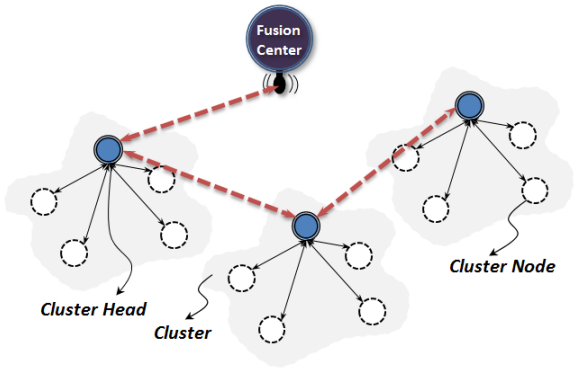

Figure 2. Communication model of the network

the epoch number $t_{i,j}$. The objective is to fit a model, $g_{net}(x,y,t|\theta)$, on the network data such that given an arbitrary location (x, y) and an epoch number t, $g(.)$ predicts the label l as the output as accurate as possible. In other words, the learning program aims to optimize the coefficients vector of the model, i.e. θ, so that the approximation error is minimized:

$$RMSE(g_{net}, n, m) = \sqrt{\frac{1}{n \times m}\sum_{i=1}^{n}\sum_{j=1}^{m}[g_{net}(x_i, y_i, t_{i,j} \mid \theta) - l_{i,j}]^2} \quad (5)$$

where m is the number of the measurements for each node. The Eq. (5) can be rewritten as [5]:

$$RMSE(g_{net}, n, m) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} E(g_i)} \quad (6)$$

where:

$$E(g_i) = \sqrt{\frac{1}{m}\sum_{j=1}^{m}[g_i(x_i, y_i, t_{i,j} \mid \theta) - l_{i,j}]^2} \quad (7)$$

In order to minimize the Eq. (5), all the network data is needed which is distributed among the nodes. In [5], every sensor node i is responsible to minimize its own $g_i(.)$ respect to its local data set. Therefore, the centralized processing required to compute the Eq. (5) can be converted into distributed processing. The

solution will be completed when some collaborations are formed between the nodes. In the gradient and Nelder-Mead Simplex based approaches, this last step is resolved by setting up a Hamiltonian path among the nodes. In the next section, we will explain a different evolutionary based approach to overcome shortcomings of these approaches.

IV. HDP Approach

Firstly, a swarm of particles is dedicated to each cluster in order to learn the relevant cluster regressor. Then, the cluster head distributes the cluster swarm among the cluster nodes and several sub-swarms are formed. Each cluster node tries to optimize its own sub-swarm through employing the PSO algorithm. Next, each cluster node sends its best particle to the corresponding cluster head. Receiving the best particles, the cluster head creates a harmony memory and runs the HS algorithm to obtain the cluster regressor. When all the cluster heads finish their improvisation processes, they collaborate to build the global model, incrementally. The HDP is described in more details in the following steps.

A. Learning the Regressors of Clusters

For the sake of simplicity and clarity, we focus on learning the regressor of a particular cluster, for example, cluster c. Let n_c and g_c denote to the size and regressor of the cluster c, respectively. Within the cluster c, the cluster nodes are required to optimize an objective function, similar to that of Eq. (5), to learn g_c as:

$$RMSE(g_c, n_c, m) = \sqrt{\frac{1}{n_c \times m}\sum_{i=1}^{n_c}\sum_{j=1}^{m}[g_c(x_i, y_i, t_{i,j} \mid \theta) - l_{i,j}]^2} \qquad (8)$$

Each cluster node $s_{c,i}$ (i=1,…,n_c) has to visit all the cluster data to compute the Eq. (8), while it has only access to its own local data set. In order to resolve this problem, re-sampling technique has been used [8]. Let each cluster node $s_{c,i}$ computes its local temporal data model using a simple curve fitting method. These temporal models can be computed by omitting the spatial part from g_c. Afterwards, all the cluster nodes send their local temporal models as well as their locations to the cluster head. Next, the cluster head broadcasts this information to all its member nodes. At this point, each cluster node regenerates those parts of the cluster data, which are not accessible for it, and appends them to its local data set. Using this technique, all the cluster nodes are given a nearly identical data view of the cluster data. Now, every cluster node will be able to compute the Eq. (8). Although the fitness (RMS error) of each particle must be evaluated based on the actual data points, the difference between the actual data and regenerated ones is trivial respect to high accuracy of the local temporal models [8]. So, instead of transmitting the actual data between the cluster nodes, which brings remarkable energy consumption, the cluster nodes can efficiently evaluate their particles based on the unified in-cluster data view. In Figure 3, this step has been shown as in-cluster data view unification.

In order to create the cluster swarm, one may forces CH_c to initialize the swarm and distribute the particles among the cluster nodes. But due to energy conservation considerations, we let CH_c send only the initial parameters to the cluster nodes. For this purpose, CH_c builds a driver message, containing two parameters PF (particle factor) and DV (domain vector), and sends the message to its member nodes.

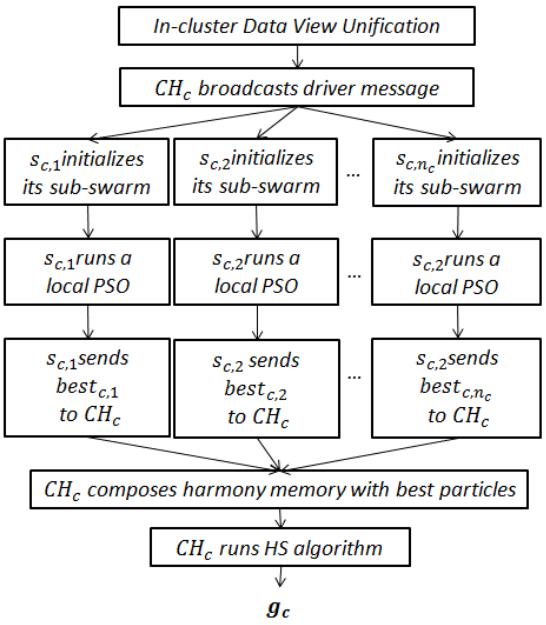

Figure 3. In-cluster optimization steps for learning one cluster regressor

The parameter PF denotes to the size of each sub-swarm and DV includes the permissible range of each dimension. Receiving the driver message, each cluster node $s_{c,i}$ creates its own sub-swarm by the size of PF and initializes its particles respect to DV. Then, $s_{c,i}$ starts its local optimization process based on the PSO algorithm until the stopping condition is satisfied. At this point, $s_{c,i}$ sends its best particle, $gbest_{c,i}$, to CH_c.

Commonly, a number of particles' migrations should be performed among the sub-swarms to give a guarantee that the sub-swarms are finally converged to the cluster regressor. But these migrations might impose undesirable communication overheads and energy consumptions. For this reason, instead of performing explicit migrations, we have employed the HS algorithm as it has almost the same functionality of the migration strategy. As pointed out in section 2, the third property of HS leads to improvise a new harmony through consideration of all the harmonies.

Accordingly, we let CH_c compose a harmony memory using the received best particles. Afterwards, the cluster head runs the HS algorithm, which can simultaneously combine and improve the best particles, to obtain the final cluster regressor, i.e. g_c. It is obvious that the size of the harmony memory, HMS, equals to the size of the cluster, n_c. The steps of in-cluster optimization processes have been shown in Figure 3. After obtaining all the clusters' regressors, the global model, g_{net} is built through the cluster heads' collaborations.

B. *Learning the Global Model*

Up to now, we have presented a bottom-up point of view from computing the local temporal models to obtaining the clusters' regressors. In order to construct the global model, it is helpful to make a top-down viewpoint. As a consequent, we will be confronted with a multiple classifier system in which the ultimate goal is to obtain a more accurate learner by combining several base learners. A set of consistent classifiers may be appeared in a system in the following ways [22]:

- Different initializations
- Different parameter choices
- Different architectures
- Different classifiers
- Different training sets
- Different feature sets

In HDP, each cluster regressor is learned based on the relevant cluster data. Thus, the presence of disjoint training data sets, one per each cluster, is the reason to have several regressors (real-valued classifiers) in our system. On the other hand, some combination rules have been proposed for combination of several base learners [23, 24]. Although choosing the best option depends on the problem at hand, some efforts have been done to make a comparison between the popular combination rules [25]. The results show that the weighted averaging rule can efficiently work in many applications, practically. In addition, it can be simply applied in a distributed environment. Accordingly, we have used the weighted averaging rule to combine the clusters' regressors to build the global model. To this end, the weight of each cluster regressor is computed based on its RMS error on the cluster data. Within the cluster c, CH_c sends its regressor g_c to the cluster nodes. The cluster node $s_{c,i}$ computes a partial RMS error $SE_{c,i}$ for g_c based on its local data set, and sends the result to the cluster head. Then, CH_c computes the final RMS error of its regressor as:

$$RMSE(g_c, n_c, m) = \sqrt{\frac{1}{n_c}\sum_{i=1}^{n_c} SE_{c,i}} \qquad (9)$$

where:

$$SE_{c,i} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}[g_c(x_{c,i}, y_{c,i}, t_{c,i,j} \mid \theta) - l_{c,i,j}]^2} \qquad (10)$$

and, the initial weight of g_c is computed as:

$$w_c = \frac{1}{RMSE(g_c)} \qquad (11)$$

These initial weights are normalized as:

$$\hat{w}_c = \frac{w_c}{\sigma} \qquad (12)$$

where:

$$\sigma = \sum_{i=1}^{C} w_i$$

Finally, the global model is computed as the linear combination of the weighted regressors:

$$g_{net} = \sum_{i=1}^{C} \hat{w}_i \times g_i \qquad (13)$$

The only remaining issue is that computing σ in Eq. (12) needs to know the RMS errors of all the clusters' regressors. Although it is possible to gather all the regressors as well as their initial weights in the fusion center, but it is more preferable to compute the equations (11)-(13), distributively, particularly when the network grows in size. The global model can simply be obtained through two traversing the cluster heads, starting from CH_1 to CH_C, and vice versa, as

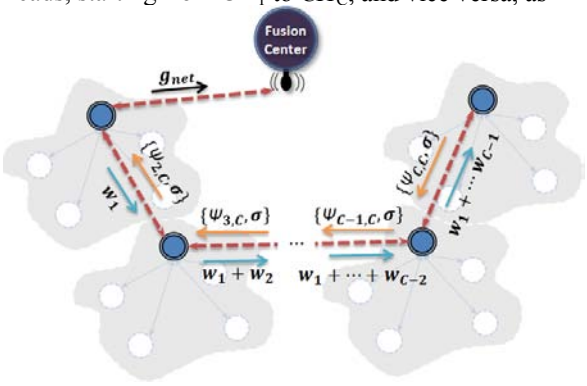

Figure 4. The cluster heads collborate to build the final network model.

shown in Figure 4. Let $\psi_{u,v}$ denote to the obtained regressor of the clusters u, u+1, …, v as:

$$\psi_{u,v} = \sum_{i=u}^{v} \hat{w}_i \times g_i \qquad (14)$$

As shown in Figure 4, at the end of the first traversing, σ is obtained by CH_C. In the second traversing, each cluster head CH_k receives σ and $\psi_{k+1,C}$ from CH_{k+1}, computes $\hat{w}_k$ and $\psi_{k,C}$ according to Eq. (12) and (14), respectively, and sends σ as well as $\psi_{k,C}$ to CH_{k-1}. In this way, the Eq. (13) is computed incrementally, and CH_1 will be able to deliver g_{net} to the fusion center.

V. EVALUATION AND EXPERIMENTS

In order to demonstrate the effectiveness of the proposed approach, we have used available Berkeley Intel Lab data set [26] which has been known as a standard data set used in many research papers in the context of WSN (e.g. [27], [29, 30]). This network consists of 54 sensor nodes with 2 corrupted ones. Sensor nodes have captured four phenomena temperature, humidity, light and voltage in each reading and expand them by date, time, epoch number, node location, and node id to compose one record. In our simulation, we have chosen node location (x, y), epoch number, and temperature from each record. It has been assumed that the network model is refreshed at the end of each day. Therefore, we have selected one-day measurements from the Berkeley data set (between Feb. 28th and Feb. 29th, 2004). So, each sensor node has 2880 data points (m = 2880), and there are totally 52×2880=149,760 records in the network in our experiments.

In the simulated network, the sensor nodes have been distributed in a two dimensional area, within the sensing field from (0, 0) to (45, 35) meters, according to their real positions [26]. The network has also been partitioned into 5 clusters as shown in Figure 5.

We have followed [27] to choose a model for fitting on the network data. They have suggested three

polynomial models, and reported their RMS errors on the Berkeley data set. It has been found that a linear space with quadratic time model has the lowest RMS error as:

$$g(x,y,t \mid \theta) = \theta_1 x + \theta_2 y + \theta_3 t^2 + \theta_4 t + \theta_5 \qquad (15)$$

Accordingly, Eq. (15) has been chosen to fit on the network data. Before the learning process starts, the

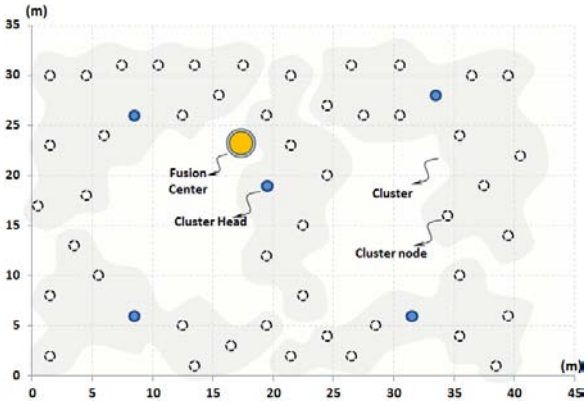

Figure 5. The simulated network

parameters of PSO and HS algorithms should be initialized by the cluster heads. The inertia weight, w, varies dynamically from w_{max} to w_{min} according to a linear decreasing strategy to allow the particles explore in the initial search steps while favoring exploitation as time increases [28]:

$$w(i) = w_{min} + \frac{p_{max} - i}{p_{max}}(w_{max} - w_{min}) \qquad (16)$$

where i and p_{max} denote to the current iteration number and the maximum number of iterations of PSO, respectively. It has experimentally been found that c_1=c_2=2.0 to be most suitable. The parameter PF is set equally for each cluster. So, each cluster's swarm has $PF \times n_c$ (c=1,...,C) particles in total. As discussed in section 4, the size of the harmony memory is set to the size of each cluster. The parameters PAR and bw are dynamically varied according to Eq. (17) and (18) to improve the performance of the HS [18]:

$$PAR(i) = PAR_{min} + (\frac{PAR_{max} - PAR_{min}}{h_{max}}) \times i \qquad (17)$$

$$bw(i) = bw_{max} \times e^{i\alpha} \qquad (18)$$

where h_{max} is the maximum number of iterations in HS and α is set as [18]:

$$\alpha = \frac{\ln(\frac{bw_{min}}{bw_{max}})}{h_{max}} \qquad (19)$$

The proposed approach has been compared with its three popular counterparts: the centralized, Incremental Gradient (IG), and Incremental Nelder-Mead Simplex (IS) approaches. The results have been analyzed in terms of the latency, prediction accuracy, and energy efficiency. All the simulations have been done using Java programming atop Eclipse environment.

A. Latency

Usually, the latency is measured based on the running time. But it is not possible to measure the latency of any WSN application in such a way. All the sensor network services can be divided into two broad categories: application layer and network layer services. The available services in the application layer are more divided into event-driven and data-centric

Table 1. The prediction accuracy comparision

Approach	Centralized	HDP	IS	IG
RMSE	**2.536**	**3.804**	**5.059**	**21.549**

applications. An event-driven application, e.g. target tracking, requires detecting and handling distinct events as well as making and keeping the exact ordering between messages. Thanks to sequential nature of the most of the event-driven applications, the latency can be exactly measured by computing the running time of the application.

In the current problem, all the discussed approaches are data-centric where the algorithms are derived based on the collected data, periodically. The efficiency of these applications depends on performing the large part of the processing in parallel. For example, in the centralized approach, most of the time is spent for transmitting the sensors readings from the sensor nodes to the fusion center, which is occurred in parallel. Also, in HDP method, all in-cluster optimizations are performed simultaneously within the clusters, as was shown in Figure 3. With regard to such parallel processing steps, the running time cannot measure the real delay of each approach for building the network model. For this reason, it is not common to measure the latency in data-centric applications in the context of WSNs ([3-10], [27]).

However, we followed from [6] to define the latency as the number of iterations to meet all the network data for the first time. For the centralized case, this happens in the first iteration of the learning program, i.e. $O(1)$. In both IG and IS algorithms, at least one network cycle must be met to visit the entire network data for the first time. This causes the latency to be in the order of the network size, i.e. $O(n)$. In HDP, the network nodes start their operations in parallel. In other words, as soon as each cluster node starts its optimization process based on the PSO, all the network data are seen. Therefore, the latency of HDP is $O(1)$ which is equivalent to the centralized one.

B. Accuracy

The RMS error of the final model has been used to show the prediction accuracy of each approach. In order to avoid overfitting problem, we have adopted 10 fold cross validation technique [31]. In the training phase, one-tenth of each sensor's local data set has been used for the test, and the network model has been learned using the remaining data points. This process has been repeated ten times and the averaged results have been reported in Table 1. As all the data points are available to the fusion center in the centralized approach, a high accurate model can be achieved. In

comparison to the centralized one, IG obtains an average accuracy, while IS brings more accurate model. As shown in Table 1, HDP clearly works better than the IG and IS approaches and obtains an acceptable accuracy in comparison with the centralized approach. This happens thanks to employing in-cluster optimizations based on the PSO and HS algorithms, and consequently the high accuracy of each cluster regressor. Also, applying a well-qualified combination rule leads to build a high accurate final model as well.

Table 2. Nine scenarios for analyzing the behavior of HDP

Scenario No.	*PF*	*HMS*
1	3	*cs*
2	3	2×*cs*
3	3	3×*cs*
4	5	*cs*
5	5	2×*cs*
6	5	3×*cs*
7	10	*cs*
8	10	2×*cs*
9	10	3×*cs*

In order to see the influence of the important parameters on HDP, nine scenarios have been defined as shown in Table 2. In each scenario, *PF* and *HMS* parameters have been set to different values. Let *cs* denotes to the cluster size in general. The parameter *PF* has been set to 3, 5, and 10 and *HMS* has been set to *cs*, 2×*cs*, and 3×*cs* in different scenarios. By setting *HMS* to 2×*cs* and 3×*cs*, we mean two and three best particles from each cluster node are sent to the corresponding cluster head, respectively.

The final RMSE of each scenario has been shown in Figure 6. It can be concluded that employing more particles and harmonies bring higher accuracy. On the other hand, creating more particles increases the complexity of computation in each sensor node. So, a trade-off should be considered between the accuracy and computational complexity. Moreover, increasing the size of the harmony memory in the cluster heads imposes some additional communication overheads, as more particles should be sent from the cluster nodes. Therefore, there has to be another trade-off between the accuracy and communication cost. These trade-offs could be resolved according to the pre-defined intervals for refreshing the network model. In other words, more expensive settings can be used when a huge number of measurements have been captured by the network. In Table 1, we have reported the accuracy of HDP according to the scenario 4.

C. *Energy Efficiency*

The total energy consumption of the sensor nodes is dominated by the amount of communications [32]. Energy efficiency of four approaches has been compared based on theoretical evaluation as well as the experimental measuring.

We have followed [5] to evaluate the complexity of data transmissions in each algorithm. Consider the case where n sensor nodes are distributed in a unit square area. Also, let L be the size of the parameter to be learned, i.e. θ. For the sake of simplicity, it will be assumed that the network is also partitioned into $\sqrt{n}$ clusters with $\sqrt{n}$ sensor nodes per cluster [6], when HDP is being evaluated. In the centralized approach, every sensor node sends all its data points to the fusion center over an average distance of $O(1)$ [5]. So, the communication requirements of this approach equals to $O(4\times m\times n\times 1)$, as shown in Table 3. In both IG and IS algorithms, the parameter θ is passed between two consecutive sensor nodes over an average distance of $O(\sqrt{\log^2 n/n})$. So, the communication requirements of IG and IS algorithms per each cycle is $O((n\text{-}1)\times L\times \sqrt{\log^2 n/n})$. In the last step, the final value of θ is sent by the last sensor node on the Hamiltonian path to the fusion center, leading to $O(L\times 1)$ communications. In HDP, there are three transmission types each of which with a different average distance as following:

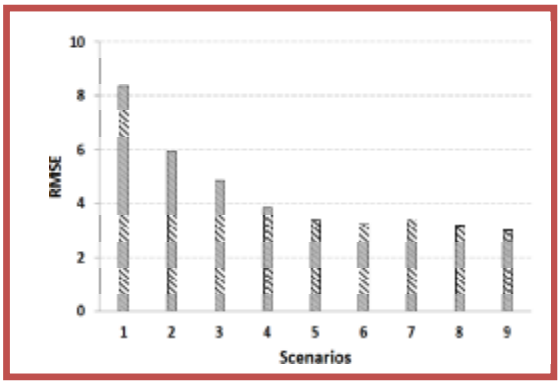

Figure 6. The influence of *PF* and *HMS* on the accuracy of HDP.

i. Transmission between a cluster head and its member nodes over an average distance of $d_1=O(1/\sqrt{n})$.

ii. Transmission between two consecutive cluster heads over an average distance of $d_2=O(\sqrt{\log^2 n/n})$.

iii. Transmission between the first cluster head and the fusion center over an average distance of $d_3=O(1)$.

Now, the communication requirements of HDP can be computed in five parts as following:

Part1. In-cluster data view unification: every cluster node transmits a vector by the size of v (node's temporal model) and two real numbers (node's location) to the cluster head in $O(\sqrt{n}\times(v+2)\times d_1)$.

Part2. The cluster head broadcasts the temporal models as well as nodes' locations to the cluster nodes in $O((v+2)\times\sqrt{n}\times d_1)$.

Part3. Learning the cluster's regressor: The cluster nodes send their best particles by the size of L to the cluster head in $O(\sqrt{n}\times L\times d_1)$.

Part4. Computing the RMS error of the cluster's regressor:

- The cluster head broadcasts its regressor to the cluster nodes in $O(1\times L\times d_1)$.

- The cluster nodes send the partial squared error of the cluster's regressor to the cluster head in $O(\sqrt{n} \times 1 \times d_1)$.

Part5. Learning the global model:

- In the first traversing the cluster heads, a real number is sent between two consecutive cluster heads in $O((\sqrt{n}-1) \times 1 \times d_2)$.

Table 3. Communication order comparison

Approach	Communication Order
Centralized	$O(n \times m \times 4 \times 1)$
IG	$O(cycles_{IG} \times (n-1) \times L \times \sqrt{\log^2 n / n}) + L \times 1)$
HDP	$O(3L + \sqrt{n}(2L+3) + (\sqrt{n}-1)(L+2)(\log\sqrt{n}/\sqrt[4]{n}$
IS	$O(Cycles_{IS} \times (n-1) \times (L+1) \times \sqrt{\log^2 n / n}) + L \times$

- Through the second traversing, a parameter by the size of L and a real number are sent between two consecutive cluster heads in $O((\sqrt{n}-1) \times (L+1) \times d_2)$.
- The first cluster head sends the network model by the size of L to the fusion center in $O(L \times d_3)$.

The total communication requirements of HDP equals to the sum of the first four parts multiplied by the number of the clusters, plus the last part as:

$$O(\sqrt{n}(Part1 + Part2 + Part3 + Part4) + Part5)$$

Table 3 compares the communication orders of four approaches. For simplicity, instead of v, its upper bound L has been taken into account. As usually $m>>L$, IG is more energy efficient than the centralized one. In practice, IG needs to meet much more cycles compared to IS to achieve an average accuracy. In our experiments, IG and IS met 40 and 1 cycles, respectively. Thus, IG consumes substantially more energy than IS. From Table 3, it can be concluded that the energy consumption of HDP falls between IG and IS approaches with (much) farther distance from IG.

The total energy consumption of four approaches has also been measured through the simulation. To do this, a simple radio energy dissipation model has been used the same as in [33]. Based on this model, energy consumed by a sensor node to transmit an l-bit message over a distance d is given by:

$$E_{TX}(l,d) = l \times E_{elec} + l \times \varepsilon_{fs} \times d^{\alpha} \qquad (20)$$

and receiving an l-bit message expends:

$$E_{RX} = l \times E_{elec} \qquad (21)$$

The parameter α is set to 2 when transmission distance is within the transmission radius of the nodes. For the experiments in this paper, the parameters for the energy consumption model have been set as [34]: E_{elec}=50 nJ/bit, ε_{fs}=10 PJ/bit/m^2, and l=1000 bits. The transmission radius of all nodes is R=15 meters. It is assumed that all the transmissions are occurred within the sensing range R, and hence α is set to 2. Also, the initial energy of each sensor node is only 2.5 J.

The total energy consumptions of four approaches have been compared in Table 4. As it is expected, the centralized and IS approaches have the highest and the lowest energy consumptions, respectively. The amount of energy consumed by HDP is about 1/14 of IG and three times more than that of IS. Of course, it should be noticed that both IG and IS algorithms require a pre-processing step to establish a Hamiltonian path between the nodes which considerably imposes an additional communication overhead. But, it has been omitted in our experiments. The network lifetime has also been shown in Table 5. We have defined the network lifetime as the maximum number of the queries that the node with the highest communications runs out of energy after that. In the centralized one, the first node is dead at the beginning of the second query. As shown in Table 5, the network lifetime of HDP is about 1/4 of the IS and 4 times more than that of IG.

Table 4. Total energy consumption of four approaches

Approach	Total Energy Consumption (J)
Centralized	17.56172
IG	0.21058
HDP	**0.01593**
IS	0.00527

Table 5. Network lifetime comparison

Approach	#. of Processed Queries
Centralized	1
IG	310
HDP	**1296**
IS	6216

VI. CONCLUSION

In this paper, a new distributed evolutionary based approach, denoted as HDP, is proposed for doing optimization-based regression in sensor networks. The proposed algorithm considers a clustered network and learns the regressor of each cluster, separately. Learning the regressor of each cluster is sponsored by a swarm of particles, initially distributed among the cluster nodes, and performing the PSO algorithm on each sub-swarm. In order to guarantee the convergence of the sub-swarms, the best of which is sent to the cluster head. Afterwards, HS algorithm is applied on the received best particles and the cluster regressor is obtained after completion of improvisation process. Finally, the global model can be obtained through applying the weighted averaging combination rule on the clusters' regressors. Thanks to cluster-level parallelism, the latency has been substantially decreased compared to IG and IS based approaches. Due to high accuracy of the clusters' regressors, and also benefiting from a well-qualified combination rule, the accuracy has been improved. The HDP obtains a model 82% and 24% more accurate than IG and IS approaches, respectively. However, in comparison with the centralized case, the accuracy should be improved even further. In practice, the energy consumption of HDP is respectively 1/1000 and 1/14

of the centralized and IG, and is three times more than that of IS approaches.

Within a cluster, each node has to manage only a few particles and the corresponding cluster head should improvise a number of harmonies proportional to the cluster size. As both PSO and HS algorithms have simple exploration mechanisms and do not use expensive mathematical operations, the computational cost of the in-cluster optimization step is not be a major concern for the limited processing capabilities of the sensor nodes. Regarding to the three discussed metrics, the proposed approach seems to be an efficient method for learning the network data model.

ACKNOWLEDGMENT

This research was supported in part by the Research Institute for ICT (ex ITRC) under Grant ITRC-17581/500.

[1] Kononenko, and M. Kukar, Machine Learning and Data Mining: Introduction to principles and algorithms, Chichester: Horwood publishing, 2007.

[2] Vehbi C. Gungor and Gerhard P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," IEEE Trans. on Industrial Electronics, vol. 56, No. 10, October 2009.

[3] Joel B. Predd, S.R. Kulkarni, and H.V. Poor, "Distributed Learning in Wireless Sensor Networks," IEEE Signal Processing Magazine, pp. 56-69, July 2006.

[4] Joel B. Predd, S.R. Kulkarni, and H.V. Poor, "A Collaborative Training Algorithm for Distributed Learning," IEEE Trans. on Information Theory, vol. 55, No. 4, April 2009.

[5] M. Rabbat and R. Nowak, "Distributed Optimization in Sensor Networks," in International Symposium on information processing in sensor networks, Berkley, California, USA: ACM, 2004.

[6] S. H. Son, M. Chiang, S. R. Kulkarni, and S. C. Schwartz, "The Value of Clustering in Distributed Estimation for Sensor Networks," in proceedings of International Conference on Wireless Networks, Communications and Mobile Computing, vol. 2, Maui, Hawaii, pp. 969–974, 2005.

[7] N. M. Charkari and P. J. Marandi, "Distributed Regression Based on Gradient Optimization in Wireless Sensor Networks," in proceedings of first Iranian Data Mining Conference, Tehran, Iran: IDMC, 2007.

[8] P. J. Marandi, N.M. Charkari, "Boosted Incremental Nelder-Mead Simplex Algorithm: Distributed Regression in Wireless Sensor Networks," in Proc. of International Conference on Mobile and Wireless Communications Networks, Toulouse, France: Springer, 2008.

[9] P. J. Marandi, M. Mansourizadeh, N. M. Charkari, "The Effect of Resampling on Incremental Nelder-Mead Simplex Algorithm: Distributed Regression over Wireless Sensor Network," in Proc. of International Conference on Wireless Algorithms, Systems and Applications, Dallas, TX, USA: Springer, 2008.

[10] H. Shakibian and N.M., Charkari, "Two-Fold Spatiotemporal Regression Modeling in Wireless Sensor Networks," in Proc. of the 6th International Conference on Advanced Data Mining and applications, LNCS, Vol. 6441, pp. 451–462, 2010.

[11] E. Alpaydm, Introduction to machine learning. Cambridge: MIT Press, 2004.

[12] A.P. Engelbrecht, Computational Intelligence: An introduction. 2nd ed., Wiley 2007.

[13] B. W. and Z. He, "Distributed Optimization over Wireless Sensor Networks using Swarm Intelligence," in Proc. Of International Symposium on Circuits and Systems, pp. 2502-2505, 2007.

[14] T. Wimalajeewa and S.K. Jayaweera, "Optimal Power Scheduling for Correlated Data Fusion in Wireless Sensor Networks Via Constrained PSO," IEEE Trans. On Wireless Communications, vol. 7, No. 9, 2008, pp. 3608-3618.

[15] S. C. Satapathy , J.V.R. Murthy, P.V.G.D. Prasad Reddy, B.B. Misra, P.K. Dash , G. Panda, "Particle swarm optimized multiple regression linear model for data classification," Elsevier, J. Applied Soft Computing, pp. 470–476, 2009.

[16] J. Behnamian, S.M.T. Fatemi Ghomi, "Development of a PSO–SA hybrid metaheuristic for a new comprehensive regression model to time-series forecasting," Elsevier, J. Expert Systems with Applications, pp. 974–984, 2010.

[17] Z. Geem, J. Kim, G. Loganathan, "A new heuristic optimization algorithm: harmony search," Simulation, vol. 76, No. 2, pp. 60–68, 2001.

[18] M. Mahdavi, M. Fesanghary, E. Damangir, "An improved harmony search algorithm for solving optimization problems," Applied Mathematics and Computation, vol. 188, pp. 1567–1579, 2007.

[19] M.G.H. Omran, M. Mahdavi, "Global-best harmony search," Elsevier, J. Appl. Math. Comput., vol. 198, 2007, pp. 643-656.

[20] R. Stoleru, T. He, J. A. Stankovic, and D. zuebke, "A high-accuracy low-cost localization system for wireless sensor networks," In Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys), 2005.

[21] A.A. Abbasi, M. Younis, "A survey on clustering algorithms for wireless sensor networks," Elsevier, J. Computer Communications, pp. 2826–2841, 2007.

[22] R. P. Duin, "The combining classifier: to Train or Not to Train?," in proceeding of 16th International Conference on Pattern Recognition, Vol. 2, pp. 765-770, 2002.

[23] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Hoboken, N.J.: Wiley, 2004.

[24] R Polikar, "Ensemble based systems in decision making," IEEE Circuits and systems magazine, vol. 6, No. 3, pp. 21-45, 2006.

[25] D. M.J. Tax, M. v. Breukelen, R. P.W. Duin, J. Kittler, "Combining multiple classifiers by averaging or by multiplying?," Elsevier, J. Pattern Recognition, pp. 1475-1485, 2000.

[26] "http://berkeley.intel-research.net/labdata/."

[27] C. Guestrin, P. Bodi, R. Thibau, M. Paskin, and S. Madde, "Distributed Regression: An Efficient Framework for Modeling Sensor Network data," in proceedings of third international symposium on Information processing in sensor networks, Berkeley, California, USA: ACM, pp. 1–10, 2004.

[28] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the IEEE International Congress on Evolutionary Computation, vol. 3, pp. 101–106, 1999.

[29] Y Le Borgne, G Bontempi, "Unsupervised and supervised compression with principal component analysis in wireless sensor networks," in Proc. Of 13th ACM International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, NY, pp. 94-103, 2007.

[30] Guorui Li, Jingsha He, Yingfang Fu, "Group-based intrusion detection system in wireless sensor networks," Elsevier J. of Computer Communications , Vol 31, pp. 4324–4332, 2008.

[31] Moore A.W., "Cross-validation for detecting and preventing overfitting, Statistical Data Mining

Tutorials, 2001, available at: http://www.autonlab.org/tutorials/.
[32] I.F. Akyildiz , W. Su, , Y. Sankarasubramaniam , E. Cayirci, “Wirelesssensornetworks: asurvey,” Elsevier J. of Computer Networks, Vol. 38, No. 4, pp. 393–422, 2002.
[33] Wendi B. Heinzelman, Anantha P. Chandrakasan, and Hari Balakrishnan, “An Application-Specific Protocol Architecture for Wireless Microsensor Networks,” IEEE Trans. On Wireless Communications, Vol. 1, No. 4, pp. 660-670, 2002.
[34] Konstantinos Kalpakis , Shilang Tang, “Maximum lifetime continuous query processing in wireless sensor networks,” Elsevier J. of Ad Hoc Networks, Vol. 8, pp. 723–741, 2010.

Hadi Shakibian received his B.Sc. degree in Computer Engineering from Arak University, Arak, Iran, and his M.Sc. degree in Software Engineering from Tarbiat Modares University, Tehran, Iran, in 2007 and 2010, respectively. His current research interests include machine learning and statistical pattern recognition, computational intelligence, and distributed and parallel computing.

Nasrollah Moghadam Charkari received hisB.Sc. degree in Computer Engineering from Shahid Beheshti University, Tehran, Iran, in 1987, and his M.Sc. and Ph.D. degrees in Computer Engineering and Information System Engineering from Yamanashi University, Japan, in 1993 and 1996, respectively.
He is currently the head of Computer Engineering Group at the Faculty of Electrical Engineering & Computer Science, Tarbiat Modares University, Tehran, Iran. His research interests include robot vision and image analysis, image mining, parallel algorithms and processing, information technology, and data hiding and water marking.