Sensor-Field Modeling Based on In-Network Data Prediction: An Efficient Strategy For Answering Complex Queries in Wireless Sensor Networks

Jose Everardo Bessa Maia  
State University of Ceara - UECE  
Fortaleza - Ceara - Brazil  
jose.maia@uece.br

Angelo Brayner  
University of Fortaleza - UNIFOR  
Fortaleza - Ceara - Brazil  
brayner@unifor.br

ABSTRACT

In this work, we present a mechanism, denoted $\text{ADAGA} - P^*$, for managing sensor-field regression models. $\text{ADAGA} - P^*$ implements an in-network data prediction mechanism in order to only transmit data which are novelties for a regression model applied by $\text{ADAGA} - P^*$. Experiments using real data have been executed to validate our approach. The results show that $\text{ADAGA} - P^*$ is quite efficient regarding communication cost and the number of executed float-point operations. In fact, the energy consumption rate to run $\text{ADAGA} - P^*$ is 15 times lower than the energy consumed by kernel distributed regression for an RMSE difference of 0.003.

Keywords

Wireless sensor networks, Query processing, Data prediction

1. INTRODUCTION

In a WSN, sensed data are passed from sensor to sensor until a special node, denoted Fusion Center (FC), is reached. A Fusion Center (also called base station) is characterized by having robust disk storage, unlimited energy power and good processing capacity. Potential WSN applications include environmental monitoring (e.g., traffic, habitat), industrial sensing and diagnostics (e.g., factory supply chains), infrastructure protection (e.g., water distribution), battlefield awareness (e.g., multitarget tracking) and context-aware computing (e.g., intelligent home) [1].

The critical point in WSNs is energy consumption [4]. The largest part of energy consumption in a sensor node occurs during data transmission or reception. For that reason, the main goal of most algorithms designed for WSN applications is communication cost reduction in terms of energy consumption [1, 2, 3, 9, 13]. For example, the use of in-network aggregation operators [2, 9, 13] is an efficient strategy to reduce the volume of data transmitted in a given WSN. However, aggregation operators do not give a faithful representation of the original structure of sensed data. To overcome this problem, an efficient strategy for reducing the data flow in a WSN is building models for sensor-field in the FC. With such models, query engines may predict and interpolate data instead of requiring them from the network.

The execution of instructions by the processing unit of sensor nodes consumes energy as well [10]. In this sense, there may exist strategies for reducing energy consumption in WSNs, which requires the execution of a huge amount of instructions (e.g., floating-point operations), in such a way that the overall energy consumption reduction becomes insignificant.

In this work, we present an engine, denoted $\text{ADAGA} - P^*$, for managing sensor-field models. The idea is to provide query engines (running upon $\text{ADAGA} - P^*$) the ability of answering complex queries in WSNs. We consider a query $Q$ as complex if, and only if, it is necessary to use sensor-field models to process it. To illustrate the notion of complex query, consider the following query $Q$: *give me the probability of rain for the next two days in the city of Fortaleza*. Roughly speaking, to process $Q$ $\text{ADAGA} - P^*$ applies regression models for predicting temperature, pressure and humidity values for the next 2 days. Afterward, it merges those models to construct a model which is able to give the rain likelihood for the next 2 days.

Another sort of applications which motivates the use of approach proposed in this paper is top-$k$ query processing. In general, a top-$k$ query returns the $k$ items having largest (or lowest) global score in a distributed system. $\text{ADAGA} - P^*$ can be applied to process top-$k$ queries, since largest or lowest values can be found from the instantiation of a sensor-field model.

The gist of $\text{ADAGA} - P^*$ is to construct sensor-field models based on regression (linear, nonlinear, kernel, etc) in regular time intervals $T_p$. The models are built to attend specific application requirements. Thus, there may be different models for a single physical phenomenon (e.g., temperature or pressure). After building an efficient model $m$, $\text{ADAGA} - P^*$ can instantiate it by applying $m$ to a given time series, generating a time sequence of sensor-field models. This way, the proposed mechanism makes query engines able to process complex queries which require forecasting, interpolation or inference.

Differently from the kernel distributed regression approach proposed in [5, 6, 7], $\text{ADAGA} - P^*$ constructs and instantiates sensor-field models in the fusion center. Such a feature saves power in sensor nodes, since they do not have to consume energy for computing the sensor-field model. The kernel distributed regression approach reduces the data volume flowing in the network by performing in-network sensor-field regression based on spatial correlation (of sensed data). Thus, only model coefficients are sent to the fusion center. Notwithstanding, for computing the sensor-field model, the number of floating-point operations (FPO) executed by kernel distributed regression is of one magnitude order higher than...
the number of FPO executed by $\text{ADAGA} - P^*$ (see Section 5).

Albeit $\text{ADAGA} - P^*$ constructs sensor-field models in the fusion center, it does avoid all sensed data to fusion center. Such a feature stems from the fact that $\text{ADAGA} - P^*$ implements an in-network data prediction mechanism. The implemented prediction mechanism relies on a linear regression model for time-series data yielded by sensors belonging to a WSN. Nevertheless, other prediction models can be used, such as polynomial regression or spline. Additionally, $\text{ADAGA} - P^*$ introduces the property of adjustable accuracy for the proposed prediction strategy. The idea is to allow users to specify a “tolerable difference” $r$ between the sensed value $s$ and the predicted value $p$.

For empirically validating the proposed approach, we have compared results produced by $\text{ADAGA} - P^*$ with those produced by kernel distributed regression [5] w.r.t. the following metrics: communication cost, reconstruction RMS error in the measuring points, and required computation (number of floating-point operations) in sensor nodes. The results show that the energy consumption reduction rate achieved by $\text{ADAGA} - P^*$ is 15 times higher than the rate obtained by kernel distributed regression for an RMSE difference of 0.003.

The remaining sections are organized as follows. Basic concepts on WSN and kernel distributed regression are presented in Section 2. In turn, Section 3 analyzes two approaches to build prediction models, where both of them use kernel distributed regression. In Section 4, we describe and discuss $\text{ADAGA} - P^*$. Section 5 presents the results obtained from experiments using real data. Finally, Section 6 concludes the paper.

## 2. BASIC CONCEPTS

Figure 1 illustrates an abstract WSN model topology assumed in this work. In this model, a WSN can be composed of three types of sensor nodes: (i) leaf nodes which are responsible for sensing, transmitting sensed data and receiving and forwarding data from other sensors; (ii) sink nodes which aggregate and predict sensed data, besides sensing data, and; (iii) a Fusion Center (or base station) which is characterized by not having battery or processing capacity restrictions.

![Figure 1: WSN topology.](image)

In our approach, users or applications submit complex queries to the query processor which is the interface between users or applications and $\text{ADAGA} - P^*$ (see Figure 1). Tasks such as data fusion, interpolation, estimation and prediction are performed by $\text{ADAGA} - P^*$ in FC. $\text{ADAGA} - P^*$ is not query-engine specific, i.e. it can be deployed on any query engine. Nonetheless, to run the experiments presented in this paper, $\text{ADAGA} - P^*$ has been used with the query engine proposed in [1], which processes queries expressed by means of Sensor Network Query Language (SNQL).

Approaches using kernel based distributed regression have been proposed, for kernel distributed regression is more efficient for extracting complete information about the sensor data shape and structure than the most aggregation schemes, while using much less communication than methods which require that every sensed data should be transmitted to FC. Next, we present some basic concepts on kernel distributed regression [5, 6, 7].

Consider a network of $N$ sensor nodes that are distributed over the interest region $\Omega$, where $\Omega \subset \mathbb{R}^2$. Each sensor measures a physical phenomenon modeled by a function in time $f(x,t)$. Let $D$ be the set of $S$ sensor measurements. Thus, the task of data modeling is to determine a function $\hat{f}(x,t)$ to fit sensor measurements in $D$.

The region $\Omega$ can be decomposed into $L$ overlapping regions $\{\Omega_j\}_{j=1}^L$, where $\bigcup_{j=1}^L \Omega_j = \Omega$. Each $\Omega_j$ can be modeled by a basis function set $H_j^f = \{h_j^f\}_{d_j=1}^k$, where $d_j$ is the number of sensors in $\Omega_j$. The local fitting function in $\Omega_j$ is defined by $\hat{f}_j(x,t) = \Sigma_{i=1}^{d_j} w_i h_i^f(x,t)$. Now, let $w_j = [w_1^j, w_2^j, ... , w_{d_j}^j]$. With this notation, the global fitting function $\hat{f}(x,t)$ is the weighted sum from all the local models:

$$\hat{f}(x,t) = \Sigma_{j=1}^L \kappa_j(x) \hat{f}_j(x,t).$$

Here, $\kappa_j$ represents the correlation degree of location $x$ associated with $\Omega_j$ and it is defined continuous and positive on $\Omega_j$ and zero elsewhere. The $\kappa_j$ are the kernel weight functions and sum to one for each $x \in \Omega$. Hence, the goal is to seek the weighted coefficients $w = [w_1^T, w_2^T, ... , w_L^T]^T$ to minimize the sum of square errors,

$$w^* = \min_w \|Hw - f\|^2,$$

where $f \in \mathbb{R}^S$ is arranging all sensor measurements into a column, and $H = [H_1, H_2, ... , H_L] \in \mathbb{R}^{S \times M}$ is an $S \times M$ matrix, where $\sum_{i=1}^{L} d_i$ and each submatrix $H_i$ is an $S \times d_i$ matrix. Each row vector $r_j$ in $H_i$ is taken from a sample point say $(x, t)$ in $D$, i.e.,

$$r_j = \kappa_j(x) \times [h_1^j(x, t_1), h_2^j(x, t_1), ..., h_{d_j}^j(x, t_1)].$$

Setting the gradient of this quadratic objective to zero gives the optimal coefficients:

$$w^* = (H^T H)^{-1} H^T f = A^{-1} b,$$

where $A = H^T H = [A_{i,j}] \in \mathbb{R}^{M \times M}$ and $b = H^T f = [b_i] \in \mathbb{R}^M$. Notice that the local models are coupled and can not be directly locally solved.

In a sequential process, such as temperature measurement over time, we need to fit a regression model using a sliding window. That is, we fit the coefficients of our basis functions regarding the measurements performed in the last $T$ minutes. The equations for this update procedure can be found in [5]. Moreover, the distributed implementation of the algorithm requires the update of partial matrices and vectors in each region using data received from the regions in which there are overlaps. The detailed equations for this update can be found in [6].

## 3. RELATED WORK

Ren and Liang present in [12] an approach for automatically selecting the WSN node for processing a given query regarding two complementary criteria: energy consumption and data quality. The idea is to probabilistically guarantees user-defined error
bound, given a confidence interval on query accuracy, while reducing the energy consumption. Although the authors claimed that their approach adaptively builds an optimal query plan, they did not present any query engine capable of building and processing query execution plans. Additionally, the paper provides no systematic empirical assessment evidencing that the proposed approach can indeed reduce the levels of energy depletion in real or simulated WSN.

In [8], the authors propose a strategy for saving energy in WSNs by exploiting data spatio-temporal correlation. Thus, the FC node partitions sensor nodes with similar measured values into clusters. Sensor nodes within a cluster are scheduled to work alternately in order to reduce energy dissipation. The proposed strategy is not scalable since it requires all nodes in the WSN to be in direct transmission range of the FC.

Guestrin et al [5] propose a distributed Gaussian Elimination algorithm to solve the regression algebraic equation. For that, an asynchronous message passing protocol and a junction tree based routing structure are employed in an in-network fashion. However, as shown in [6] there are several drawbacks in this approach. Firstly, only one hop for asynchronous message exchanging among neighboring nodes will inevitably result in slow convergence. Secondly, the network topology is organized as a large flat structure. For that reason, it will be much costly for reconfiguring the global junction tree if some links fail in unstable networks. Finally, since several nodes do not participate for solving the regression problem, there is redundant communication activity, especially, for those nodes with the same associated clique and factor set.

To overcome the aforementioned drawbacks presented by the approach proposed by Guestrin et al [5], Hou et al [6, 7] introduce the idea of using a skeleton tree for processing the distributed Gaussian elimination algorithm. The distributed Gaussian elimination algorithm is applied to solve regression algebraic equations. The key idea behind the notion of skeleton tree is to combine the clustering based routing mechanism with message passing protocol based on junction tree.

Compared with the strategy of building a junction tree over all nodes [5], the skeleton-tree based routing protocol suppresses the redundant message exchanging among the ramose nodes and cluster heads without deteriorating data modeling precision. In [7, 6], the authors show that the routing scheme proposed by them can save about 33% communication cost w.r.t. the approach proposed by Guestrin et al.

On the other hand, the approaches proposed by Guestrin et al [5] and Hou et al [6, 7] present the same energy consumption rate regardless the sensor field varies rapidly or not. In this sense, they propose as future work an adaptive mechanism for adjusting system parameters such as region scope, time window length and the number of basis functions or kernels. Guestrin et al and Hou et al argue that such an adaptive mechanism make their approaches able to take advantage whenever the sensor field varies more slowly in order to reduce the energy consumption rate by decreasing the amount of FPOs and message exchange.

Notwithstanding, the approaches presented in [5] and [7] still require the execution of an amount of FPOs which is of one magnitude order higher than the number of FPOs executed by ADAGA – $P^*$. ADAGA – $P^*$ naturally incorporates the feature of taking advantage whenever the sensor field is varying more slowly. For instance, whenever the sensor field temperatures varies smoothly for a long period, the prediction model becomes more efficient (i.e., yields more precise predictions). Consequently, ADAGA – $P^*$ automatically transmits less data.

To conclude this section, we want to emphasize that the in-network prediction strategy proposed in this paper is more efficient than the aforementioned approaches w.r.t. energy consumption. This is because: (i) message exchange among nodes is not necessary, (ii) the amount of FPOs executed at each sensor node is significantly smaller, and; (iii) the data volume injected into network is automatically reduced case the sensor field varies more smoothly.

4. ANSWERING COMPLEX QUERIES IN WSNs

In this section, ADAGA – $P^*$ engine as well its prediction mechanism are described.

4.1 ADAGA – $P^*$: An Engine for Managing Sensor-field Models

In order to process complex queries in WSNs, we propose a mechanism, called Complex-Query Engine (for short, CQE), which is composed of a query engine and ADAGA – $P^*$ engine. Thus, the proposed CQE may process conventional and complex queries in WSNs. For this work we have used the query engine proposed in [1] to implement CQE. However, ADAGA – $P^*$ can be integrated with any other query engine, which is able to process queries in WSNs. Figure 2 depicts an abstract model for the structure of the proposed CQE.

The ADAGA – $P^*$ engine implements two key functions:

- **Sensor-field model building.** Regression-based sensor-field models are built to attend application-specific requirements. In this sense, it is necessary to have a model database. This way, users may pick a model up, which is more adequate to their applications. For example, there exist classes of applications for which simple linear regression model is enough;

- **Model instantiation.** This function has the goal of providing to query engines data produced by applying a sensor-field model (from the regression-model database).

Consequently, CQE should manage a regression-based model database, called RM database, and an instance database, denoted IRM database (see Figure 2). The RM database has the functionality of storing regression models. For example, the RM database may store linear regression models and kernel-based regression models. In turn, the IRM database stores instances for the models in the RM database. In other words, the IRM database stores data and parameters produced by applying a regression-based model to execute a complex query for a given application (or user).

To illustrate the usage of the RM and IRM databases, consider that the RM database stores a linear regression model, called LR, and a kernel based distributed regression model, denoted KDR. Thus, for a given data series $t$, the IRM database may store: (i) an instantiation $L_i$ produced by applying the linear regression model LR on the data series $t$, and; (ii) an instantiation $K_i$ yielded by applying the kernel distributed regression model KDR on $t$.

ADAGA – $P^*$ constructs and instantiates sensor-field models in the fusion center (FC). Moreover, in our approach, the number of data injected into the network is reduced. This is because ADAGA – $P^*$ implements an in-network data prediction mechanism based on temporal correlation of sensed data. As a result, data which can be predicted by the FC within the threshold error are not transmitted (see Section 4.2).

Therefore, we claim that ADAGA – $P^*$ is more efficient than the distributed regression approaches proposed in [5, 6, 7] w.r.t. energy consumption diminution rate. The results presented in Section 5 prove that our claim is true.
In the proposed prediction strategy, the linear regression equation (prediction equation) is sent to all leaf nodes belonging to a WSN. It is important to observe that, if a leaf node does not receive the prediction coefficients, it has a chance to receive the coefficients values again when it sends a new set of novelties to the sink node.

The in-network prediction implemented by $\text{ADAGA} - P^*$ is based on the following equation: $\hat{S}(t) = a + bt$. The time $t$ is an independent variable. $\hat{S}(t)$ represents the estimated value of $S(t)$ and is variable with $t$. Parameter $a$ is the intercept-$t$ (value of $\hat{S}(t)$ for $t = 0$) and $b$ is the stretch slope, and are computed as follows:

$$a = \frac{1}{N} \left( \sum S_i - b \sum t_i \right) = \bar{S} - b\bar{t},$$

$$b = \frac{\sum (t_i - \bar{t})(S_i - \bar{S})}{\sum (t_i - \bar{t})^2}.$$  

$\text{ADAGA} - P^*$'s prediction mechanism requires that each sink node (see Figure 1) executes the steps described below and in the following chronological order:

1. The sink node receives data from sensor nodes in its region to compute the coefficients $a$ and $b$.
2. Each sink node sends the coefficients to all sensors belonging to its region. In turn, sensor nodes start to compare collected data with predicted data, checking the threshold error. A sensor transmits data only when the absolute difference between the sensed value and the prediction value is greater than the value specified in the threshold clause; and
3. Since each sink node is responsible for a sub-network (region) of sensors, if any value sent by a sensor reaches a sink node, the latter uses the received value and recalculates regression equation coefficients. Thus, prediction accuracy is automatically and continuously adjusted.

Figure 3 illustrates the proposed prediction mechanism behavior for the query depicted in Table 1. To plot the curves depicted in Figure 3, a set of 38 real temperature values (collected by a sensor node) has been used, with a threshold of 5%.

4.2 In-Network Data Prediction

The $\text{ADAGA} - P^*$'s in-network prediction mechanism requires that sensor nodes know the limit in the difference between the acquired value and the estimated value. To dynamically “inform” a sensor node about that difference, a new clause was added to SNQL [1], called threshold (see Table 2) [11]. The sensor does not inject sensed data into the network if the threshold value is not violated. Thus, it is enough to use the following formula: $P > 100(Vc - Vp)/Vc$, where $P$ represents the percentage value specified in the threshold clause, $Vc$ corresponds to the value collected by the sensor node, and $Vp$ is the value resulting from the prediction process (predicted value).

$$a = \frac{1}{N} \left( \sum S_i - b \sum t_i \right) = \bar{S} - b\bar{t},$$

$$b = \frac{\sum (t_i - \bar{t})(S_i - \bar{S})}{\sum (t_i - \bar{t})^2}.$$  

Therefore, in $\text{ADAGA} - P^*$ data accuracy can be dynamically specified by users through the threshold SNQL clause. The idea is to give to users/applications the decision of reducing even more energy consumption. The threshold parameter is defined for each query submitted to $\text{ADAGA} - P^*$. We are assuming that the user is a specialist and for that reason he/she has the expertise to decide for the best threshold value having in mind that for smaller threshold values more energy is consumed and more precise the results are.

In the proposed prediction strategy, the linear regression equation (prediction equation) is sent to all leaf nodes belonging to a WSN. It is important to observe that, if a leaf node does not receive the prediction coefficients, it has a chance to receive the coefficients values again when it sends a new set of novelties to the sink node.

The in-network prediction implemented by $\text{ADAGA} - P^*$ is based on the following equation: $\hat{S}(t) = a + bt$. The time $t$ is an independent variable. $\hat{S}(t)$ represents the estimated value of $S(t)$ and is variable with $t$. Parameter $a$ is the intercept-$t$ (value of $\hat{S}(t)$ for $t = 0$) and $b$ is the stretch slope, and are computed as follows:

$$a = \frac{1}{N} \left( \sum S_i - b \sum t_i \right) = \bar{S} - b\bar{t},$$

$$b = \frac{\sum (t_i - \bar{t})(S_i - \bar{S})}{\sum (t_i - \bar{t})^2}.$$  

$\text{ADAGA} - P^*$’s prediction mechanism requires that each sink node (see Figure 1) executes the steps described below and in the following chronological order:

1. The sink node receives data from sensor nodes in its region to compute the coefficients $a$ and $b$.
2. Each sink node sends the coefficients to all sensors belonging to its region. In turn, sensor nodes start to compare collected data with predicted data, checking the threshold error. A sensor transmits data only when the absolute difference between the sensed value and the prediction value is greater than the value specified in the threshold clause; and
3. Since each sink node is responsible for a sub-network (region) of sensors, if any value sent by a sensor reaches a sink node, the latter uses the received value and recalculates regression equation coefficients. Thus, prediction accuracy is automatically and continuously adjusted.

Figure 3 illustrates the proposed prediction mechanism behavior for the query depicted in Table 1. To plot the curves depicted in Figure 3, a set of 38 real temperature values (collected by a sensor node) has been used, with a threshold of 5%.
mitted to the respective sink node, which in turn recomputes the regression equation coefficients.

Therefore, from the 35 temperature values collected by a sensor node, only three were actually transmitted, corresponding to approximately 90% of communication economy. Obviously, energy economy depends, among other factors, on data sample rate or on the used prediction model.

The in-network data prediction implemented by ADAGA – $P^*$ is executed in eight distinct stages. Moreover, three different storage areas are required: (i) reception area used to store received packets (from other sensor nodes); (ii) processing area, which stores data to be processed (for example, by the prediction mechanism); and (iii) forwarding area, which temporally stores data to be transmitted later [11].

5. EVALUATION

To carry out the tests, we have considered that all sensor nodes initially had the same amount of power and available memory. The query used to generate the evaluation results is the one depicted in Table 1. We have used real data on temperature (in degree Celsius) obtained in the Meteorology and Hydric Resources Foundation of the State of Ceará (FUNCEME) in Brazil.

The data were collected by a set of 26 sensors geographically distributed in the State of Ceará (in Brazil). Each of the 26 sensors collected data every one hour (sense interval clause - see Table 1) over an interval of 46 consecutive days, which corresponds to 1,104 sensed data per sensor, totaling 28,704 collected data. Figure 4 shows a Ceará map with the location of 26 sensors. Those sensors are located in three different sub-regions (rectangles) with different climatic profile and consequently with expressive variation of temperature values: coastal, mountain and caatinga (forest composed of stunted trees and thorny bushes, found in areas of small rainfall in northeastern Brazil).

To evaluate the RMS error we have compared the surface generated by Distributed Regression method described in Section 3 with the surface yielded by ADAGA – $P^*$ in the FC. For that, we have applied the same parametric linear regression method. In this case, regression performs interpolation on all sensor data, real or predicted. That is, the base functions extend for the entire data space without the kernel restrictions. For evaluating the communication cost, the average number of messages sent or resent by sensor nodes has been computed. Finally, to compute the number of instructions executed by the sensors for generating the regression surface, we have considered the average number of floating point operations (FPOs).

In both experiments (distributed regression and ADAGA – $P^*$) the basis functions set used was the linear-space in each snapshot, as in [5]: $\tilde{f}(x, y, t) = c_1x + c_2y$.

Regarding ADAGA – $P^*$, we have used 3 different thresholds (1%, 2% and 5%) for computing in-network data prediction (see Section 4.2). The sliding window used by ADAGA – $P^*$ for data prediction was of size 7. The results are shown in Table 3. To compute communication cost, we have considered the average number of hops (denoted $\bar{n}_h$) necessary to carry a message from each sensor to the FC.

The results in Table 3 are for a small WSN where all nodes reach the FC in one hop ($\bar{n}_h = 1$). However, variations in $\bar{n}_h$ do not change the relative position of the total energy consumption among the evaluated approaches, since communication cost for all approaches is computed based only on $\bar{n}_h$. The model used for simulating energy consumption considers consumption of 75 nJ per bit transmitted, 50 nJ per bit received, and a consumption of 125 nJ/bit for sensing and processing [1].

Table 3 shows that the energy consumption to run the distributed regression was 15 times higher than the energy consumed to execute ADAGA – $P^*$ (with threshold of 1%), for an RMSE difference of 0.003.

The number of FPOs in Table 3 was calculated as follows. For ADAGA – $P^*$, we have considered 6 FPOs to predict and to compare the predicted data with the sensed data in each sensor. On the other hand, for the distributed regression approach, we have considered the following computation: (i) for each sensor in a kernel, the computation of a linear system by Gaussian elimination; (ii) $5kk$ matrices multiplication (product), where $k$ is the number of sensors in the kernel region.

We have also evaluated the impact of the threshold parameter used by ADAGA – $P^*$ to predict sensed data. For that we have run the same query with 10 different values for the threshold clause. The values are the integers belonging to the interval [1,10]. On the other hand, threshold value equal to 0 means that ADAGA – $P^*$’s in-network prediction is not used.

Network energy consumption was also verified when the in-network data prediction strategy implemented by ADAGA – $P^*$ is triggered. Figure 5 shows energy consumption of the WSN previously described. It is worthwhile to note that we have considered the both directions of transmission, i.e. from and into sensor nodes, for computing communication cost. We have computed the energy consumption of each of the 26 sensor nodes in three different scenarios: (i) when the in-network prediction is used (i.e., with in-network aggregation and prediction), (ii) when only ADAGA [2] is used (i.e., with only in-network data aggregation) and; (iii) when a naive strategy is used (i.e., without data aggregation and prediction). To plot ADAGA – $P^*$’s energy consumption curves (see Figure 5), we have considered three different threshold values: 1%, 5% and 10%.

![Figure 4: Ceará Map.](image)

Figure 4: Ceará Map.

To evaluate the RMS error we have compared the surface generated by Distributed Regression method described in Section 3 with the surface yielded by ADAGA – $P^*$ in the FC. For that, we have applied the same parametric linear regression method. In this case, regression performs interpolation on all sensor data, real or predicted. That is, the base functions extend for the entire data space without the kernel restrictions. For evaluating the communication cost, the average number of messages sent or resent by sensor nodes has been computed. Finally, to compute the number of instructions executed by the sensors for generating the regression surface, we have considered the average number of floating point operations (FPOs).

In both experiments (distributed regression and ADAGA – $P^*$) the basis functions set used was the linear-space in each snapshot, as in [5]: $\tilde{f}(x, y, t) = c_1x + c_2y$.

Regarding ADAGA – $P^*$, we have used 3 different thresholds (1%, 2% and 5%) for computing in-network data prediction (see Section 4.2). The sliding window used by ADAGA – $P^*$ for data prediction was of size 7. The results are shown in Table 3. To compute communication cost, we have considered the average number of hops (denoted $\bar{n}_h$) necessary to carry a message from each sensor to the FC.

The results in Table 3 are for a small WSN where all nodes reach the FC in one hop ($\bar{n}_h = 1$). However, variations in $\bar{n}_h$ do not change the relative position of the total energy consumption among the evaluated approaches, since communication cost for all approaches is computed based only on $\bar{n}_h$. The model used for simulating energy consumption considers consumption of 75 nJ per bit transmitted, 50 nJ per bit received, and a consumption of 125 nJ/bit for sensing and processing [1].

Table 3 shows that the energy consumption to run the distributed regression was 15 times higher than the energy consumed to execute ADAGA – $P^*$ (with threshold of 1%), for an RMSE difference of 0.003.

The number of FPOs in Table 3 was calculated as follows. For ADAGA – $P^*$, we have considered 6 FPOs to predict and to compare the predicted data with the sensed data in each sensor. On the other hand, for the distributed regression approach, we have considered the following computation: (i) for each sensor in a kernel, the computation of a linear system by Gaussian elimination; (ii) $5kk$ matrices multiplication (product), where $k$ is the number of sensors in the kernel region.

We have also evaluated the impact of the threshold parameter used by ADAGA – $P^*$ to predict sensed data. For that we have run the same query with 10 different values for the threshold clause. The values are the integers belonging to the interval [1,10]. On the other hand, threshold value equal to 0 means that ADAGA – $P^*$’s in-network prediction is not used.

Network energy consumption was also verified when the in-network data prediction strategy implemented by ADAGA – $P^*$ is triggered. Figure 5 shows energy consumption of the WSN previously described. It is worthwhile to note that we have considered the both directions of transmission, i.e. from and into sensor nodes, for computing communication cost. We have computed the energy consumption of each of the 26 sensor nodes in three different scenarios: (i) when the in-network prediction is used (i.e., with in-network aggregation and prediction), (ii) when only ADAGA [2] is used (i.e., with only in-network data aggregation) and; (iii) when a naive strategy is used (i.e., without data aggregation and prediction). To plot ADAGA – $P^*$’s energy consumption curves (see Figure 5), we have considered three different threshold values: 1%, 5% and 10%.

![Figure 5: Energy consumption.](image)

Figure 5: Energy consumption.

The potential packets loss in WSNs has a quite limited impact in the results produced by ADAGA – $P^*$. This happens because ADAGA – $P^*$ builds and instantiates the regression models using predicted values. Such values may be outside the threshold error (used by ADAGA – $P^*$’s in-network prediction). However, the ef-
fect will be diluted by surface regression unless values sent by mul-
tiple sensors are lost simultaneously.

Of course, ADAGA – P* needs to cope with the sensor failure problem. In fact, this problem is more complex in ADAGA – P*, since a sensor may stay a long time without transmitting any data, when a large set of correct predictions is yielded. In this case, two approaches may be applied. One is to establish a validity period for the prediction model, i.e., the sensor always transmits after this interval. In this case, long intervals will delay the failure detection event and shorter intervals may impact on communication cost. The other strategy is to run a process in FC, which uses the spatial correlation among sensor data to detect when significant changes may have occurred.

In [5], in order to analyze scalability in the kernel distributed regression, the authors use as metric the number of messages sent to the FC. Thus, one can argue that in distributed regression [5, 6, 7], the number of bytes sent to the FC grows linearly with the number of nodes in the network. However, it is important to observe that the number of bytes sent to the FC grows linearly with the number of WSN nodes in the kernel distributed regression if and only if the number of bytes sent to the FC always grows linearly with the number of nodes in the network. This is because ADAGA – P* requires that each sensor node just sends a subset of sensed data.

The effectiveness of simple linear regression used by ADAGA – P* for in the in-network data prediction can be substantially improved by using a more complex prediction model.

6. CONCLUSION

In this work, we described ADAGA – P*, an engine for managing sensor-field models. The idea is to provide to query engines for WSNs (running upon ADAGA – P*) the ability of answering complex queries. ADAGA – P*’s main functionalities are to construct and instantiate sensor-field models based on linear regression. The sensor-field models are built to match application specific requirements.

By means of experiments on real data, we have validated ADAGA – P*’s efficiency. The results have shown that significant gains in terms of energy savings are achieved. The results presented in Table 3 proves this assertion, since energy consumption to run kernel distributed regression was 15 times higher than the energy consumed to execute ADAGA – P* (with threshold of 1%), for a RMSE difference of 0.003.

Finally, we emphasize the applicability and the effectiveness of ADAGA – P* for a wide variety of applications which consume data produced by WSNs.

7. REFERENCES


Table 3: Evaluation results of the algorithms - total energy cost for a sensor-field snapshot ($\eta_0 = 1$).

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>Comm. cost</th>
<th>FPOs</th>
<th>Sense cost</th>
<th>Energy consumption (nJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N = 26 sensors)</td>
<td></td>
<td>($X_1$)</td>
<td></td>
<td>($X_2$)</td>
<td>($X_3 = N \times 100nJ$)</td>
</tr>
<tr>
<td>Dist. Reg.</td>
<td>0.011</td>
<td>254 + 28$\eta_h$</td>
<td>7050</td>
<td>2600</td>
<td>80100</td>
</tr>
<tr>
<td>$ADAGA - P^*$ with In-net Pred.(1%)</td>
<td>0.014</td>
<td>14$\eta_h$</td>
<td>156</td>
<td>2600</td>
<td>5130</td>
</tr>
<tr>
<td>$ADAGA - P^*$ with In-net Pred.(2%)</td>
<td>0.022</td>
<td>11$\eta_h$</td>
<td>156</td>
<td>2600</td>
<td>4755</td>
</tr>
<tr>
<td>$ADAGA - P^*$ with In-net Pred.(3%)</td>
<td>0.016</td>
<td>7$\eta_h$</td>
<td>156</td>
<td>2600</td>
<td>4255</td>
</tr>
</tbody>
</table>