# RADAR - Regression Based Energy-Aware DAta Reduction in WSN: Application to Smart Grids

Bashar Chreim, Jad Nassar, Carol Habib

# RADAR - Regression based energy-Aware DAta Reduction in WSN: Application to Smart Grids

Bashar Chreim, Jad Nassar and Carol Habib

**Abstract** The evolution towards Smart Grids (SGs) represents an important opportunity for the energy industry. It is characterized by the integration of renewable and alternative energy resources into the existing power grids while ensuring a fine-grained control for the different measuring points. Therefore, this evolution requires the ability to send a maximum of data over the network in real time while controlling the grid. A Wireless Sensor Network (WSN) deployed across the grid is a potent solution to achieve this task. However, sensor nodes have limited energy and computation resources especially the battery powered ones. For that, reducing transmission is an essential priority in order to increase the lifetime of the network. Data prediction is a widely used, yet effective, solution in literature to accomplish this task. In this paper, we propose a Quality of Service (QoS) aware algorithm based on time series prediction and linear regression for data prediction in WSN. We test our approach in a SG context on real data traces of photo-voltaic cells. Our algorithm takes into consideration the diversity of applications of SGs with different requirements while being energy efficient. Our results show that our proposal provides satisfactory results compared to literature solutions in terms of data reduction percentage, Root Mean Square Error (RMSE) and energy consumption.

**Key words:** Smart Grids, Wireless Sensor Networks, Data Reduction, Data Prediction, Linear Regression, Linear Correlation

Bashar Chreim
JUNIA, Lille, France , e-mail: Bashar.chreim@junia.com

Jad Nassar
JUNIA, Lille, France, e-mail: Jad.nassar@junia.com

Carol Habib
JUNIA, Lille, France e-mail: Carol.habib@junia.com

## 1 Introduction

The evolution towards Smart Grids (SGs) represents an important opportunity to shift the energy industry into a new era of reliability, availability and efficiency [1]. This transformation will offer huge advantages for the stake holders by giving them a broader vision, management and control of the grid while lowering the costs, as well as to the customers, making their daily life more comfortable and convenient [2]. This evolution is manifested by (1) the integration of renewable energy resources all over the grid, (2) a two-way communication between the utility and the customers and (3) automated decisions of the smart connected devices. Therefore, these changes require the ability to transmit in real time a maximum of data over the network, in order to monitor and control the different heterogeneous decentralized energy resources. A WSN deployed all over the grid on the different measuring and control points, is a potential and plausible solution to be used with SGs [1]. Moreover, in a SG, electricity and energy do exist, but connecting sensor nodes to such high voltage with intermittent and ill-adapted energy levels is sometimes inappropriate or physically impossible. For that, battery-powered sensor nodes must be deployed all over the grid, in order to ensure data transmission from source to destination node.

In a WSN, monitoring, processing and transmitting are the main tasks accomplished by sensor nodes. These sensors have limited energy and computation resources [3]. Each time interval, source nodes perform data sampling and transmission to the destination, via a set of sensor nodes distributed across the network. However, most of the time, sensed values do not change significantly between consecutive readings. This is all true to some SG applications as well (i.e., photo-voltaic cells monitoring). Sending these samples periodically will cause exhaustion of the batteries of sensor nodes (knowing that wireless communication is considered the major energy consumer [3]) and information redundancy at the destination. For that, and in order to maximize sensor nodes lifetime, data reduction has proven to be a potent solution. This is done by reducing the data transmission rate or aggregating data packets within the network.

In literature, data reduction techniques have been widely used for WSN applications [4]. However these techniques are limited to specific applications. This is mainly due to their parameters (e.g., filter length, step size) that need to be tuned to adapt to a particular data type (e.g., temperature, humidity). Therefore, these techniques need specific customization before being used in SG applications which are characterized by their heterogeneity in terms of QoS requirements and data types [5]. For the rest of the paper, we will refer to data type by variable.

In this paper, we propose *RADAR*, an energy efficient data reduction algorithm based on data prediction. It uses time series prediction and linear regression models. We note that in a previous work [6], we presented the concept of our algorithm. More precisely, our algorithm exploits correlations among different variables collected from photo-voltaic cells and creates automatically a prediction model for each variable. A time series prediction model is used to predict one of these variables, and the rest of them are predicted using linear regression models. To the best

of our knowledge, our work is the first effort to apply linear regression in data reduction for WSNs combined with a time series prediction model.

*RADAR* is tested on real data traces obtained from photo-voltaic cells. Simulation results show that our approach provides satisfactory results compared to literature solutions in terms of energy efficiency and *RMSE*, while reducing data transmission in the wireless network.

The rest of the paper is organized as follows: Section 2 presents a summary of related work. Section 3 describes our proposed solution. Section 4 shows the simulation setup and environment used to validate our proposition. Section 5 describes the performance evaluation of our approach and remaining issues are discussed in section 6. Finally, section 7 concludes the paper.

## 2 Related Work

Data reduction techniques can be divided into three main categories [4]: data compression, In-network processing and data prediction (Figure 1). The main idea behind data compression is that processing data consumes much less power than transmitting it [7]. For that, data is compressed/aggregated before leaving the source node using compression and aggregation techniques [7, 8]. Unlike data compression, the process of aggregation in In-network processing is applied at intermediate nodes between the source and the destination [9–11]. Therefore, the amount of data is reduced over the network while traversing towards the destination. In data prediction, the amount of data transmitted by the sensor nodes is reduced by predicting the sensed values using specific models. Usually, the network maintain two instances of each prediction model, one residing at the source node and the other at the destination node, this is called Dual Prediction Scheme (*DPS*) [12]. It consists of running an instance of the model on both the source and destination nodes. The destination node will start answering queries using the values predicted by its model, without any communication with the source node. However, if the difference between the predicted and the sensed value is greater than a certain threshold, the source node will send its sensed value to the destination. In the current work, data prediction is the point of our interest.

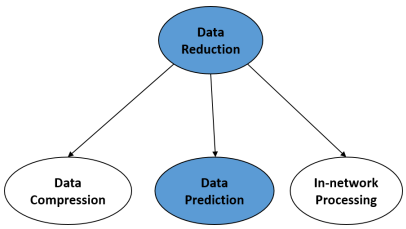


Fig. 1: Classification of energy saving in sensor networks

In literature, numerous studies focused on developing data prediction techniques in WSNs. In [13], the authors used Auto Regressive Integrated Moving Average (*ARIMA*) model for the information collection scheme. In their approach, the sink node uses historical data to build up an appropriate time series prediction model for each sensor node and send back its parameters to build the same model. In [14], the authors proposed a prediction scheme using Seasonal *ARIMA* (*SARIMA*) model for short term prediction that can predict using only limited input data. This model is used to predict traffic flow. The main drawback with these methods is their requirement of high memory and computational overhead to initially build the model and to re-compute it when outdated. In [15], Least Mean Square (*LMS*) algorithm is used for data prediction in WSN. It consists of running two instances of the model at the sensor and the sink node, applying dual prediction scheme (*DPS*). The main complexity when using *LMS* is the task of choosing the best parameters to fit to a specific data type. In [16], the authors proposed a modification for the *LMS* algorithm by adding a phase of initialization and parameters determination. They varied the algorithm parameters within specific intervals, and selected those that minimize the *RMSE*. All described techniques have been successfully used in WSN applications, but it is important to note that for each type of application, the parameters must be computed. In addition, for the same variable, parameters may not fit for all different QoS necessities. For example, for different temperature based applications (e.g., solar power forecasting, solar irradiance prediction), the determined model parameters (e.g., filter length, step size) may not fit to all these applications. In fact, this is mainly due to having several difference thresholds between sensed and predicted values based on the application needs [16].

Moreover, other work focused on linear regression for data prediction [17–19]. In [18], the authors used multiple linear regression (*MLR*) to predict solar intensity from a set of weather metrics. In [19], the authors determined the most influential features for predicting photo-voltaic production, using linear regression. However, in these work, all input variables need to be available all the time in order to predict a single output. In other words, sensor nodes that are responsible of collecting the value of these variables need to execute the transmission task all the time, which may exhaust the batteries of the sensor nodes after a period of time.

## 3 Proposed Solution

The purpose of our contribution is to create an autonomous prediction algorithm (*RADAR*) for heterogeneous applications (e.g., photo-voltaic cells monitoring, electric vehicles control) in WSN. It generates simultaneous prediction models for all variables by exploiting linear correlation among them. This is done using two types of models: *time series prediction* and *linear regression*. These models provide satisfactory and accurate results while being simple and lightweight which is the most beneficial in energy limited WSNs.

The idea is to predict the first selected variable (from a set of pre-identified vari-

ables) using a time series prediction model. Here, the prediction algorithm could be any time series model from the state of the art. In our case we used *SARIMA* as the benchmark model. After that, this variable will allow us to predict the remaining ones using linear regression models (simple linear regression (*SLR*) for the first iteration followed by *MLR* for the remainders). All of that is based on the linear correlation between every couple of variables. As already mentioned, by variable we mean the different data types (e.g., temperature, humidity) that we used in our simulation. In the rest of this section, we will detail the steps of *RADAR* that is represented in algorithm 1. We will consider an example of variables from Table 1 to support our explanation.

Table 1: Correlation matrix showing correlations between variables

| | **Temperature** | **Humidity** | **Irradiance** | **Current BP** |
|---|---|---|---|---|
| **Temperature** | 1 | -0.8154 | +0.7748 | +0.6791 |
| **Humidity** | -0.8154 | 1 | -0.7906 | -0.6673 |
| **Irradiance** | +0.7748 | -0.7906 | 1 | +0.7464 |
| **Current BP** | +0.6791 | +0.6673 | +0.7464 | 1 |

- Our algorithm takes as input a dataset with a random number of variables (e.g., temperature, humidity).
- In the first step, the correlation matrix (*CM*) is built (line 2 in Algorithm 1). It represents the correlation coefficient of each couple of variables (Table 1).
- After that, the negative values are replaced by their absolute ones in the *CM* (since the positive and negative values have the same impact on the correlation coefficients).
- In the next step, the maximum value (*Max*) in the matrix must be identified (line 8 in Algorithm 1). It represents the correlation coefficient of the most correlated couple of variables (e.g., the temperature and the humidity in Table 1 with the value -0.8154).
- A time series prediction model (Model 1 in Table 2) is created using *SARIMA* [14] for one variable of the couple, randomly selected (the temperature per example in this case). This variable will be predicted at the destination based on the *SARIMA* model.
- The output of this time series model is used as input to create an *SLR* model (line 11 in Algorithm 1), using the formula presented in [20]. It predicts the second variable previously identified. Since we chose the temperature as a random variable in the previous step, the humidity will then be the output of the current model (Model 2 in Table 2).
- The next model (Model 3) should be created now by identifying the next maximum value in the matrix (in Table 1). It is an *MLR* model, and takes as input the outputs of the previous created models (line 17 in Algorithm 1).

- This final step is repeated until prediction models are created for all variables (Model 4 in our example is the last model). Our algorithm returns a matrix that represents the input/s and output variable/s of each created model.
- Once the execution of the algorithm finishes, all the prediction models are created. The prediction process can now take place.

ALGORITHM 1 : Data prediction algorithm

**Require:** Dataset
**Ensure:** Prediction Models
1: $Count \leftarrow NbOfVariables(dataset)$ //Number of needed models
2: $CM \leftarrow BuildCorrelationMatrix()$
3: **for** each value in CM **do**
4: **if** $value < 0$ **then**
5: $value \leftarrow |value|$
6: **end if**
7: **end for**
8: $Max \leftarrow CM.MaximumValue()$
9: $X, Y \leftarrow Max.IndexInCM()$
10: $SARIMA(X)$
11: $SLR(X, Y)$
12: $Count \leftarrow Count - 2$
13: $Max \leftarrow 0$ //Replace maximum by zero in the matrix
14: **repeat**
15: $Max \leftarrow CM.MaximumValue()$
16: $X, Y \leftarrow Max.IndexInCM()$
17: $MLR(X\ and\ all\ previous\ outputs, Y)$
18: $Max \leftarrow 0$
19: $Count \leftarrow Count - 1$
20: **until** $Count == 0$ //Each variable has its prediction model

In a nutshell, *RADAR* works as follows: a time series prediction model will predict the upcoming value of one of the variables of the applications. Next, the predicted value is used as input by an *SLR* model to predict the value of the second variable. Then, *MLR* models are executed simultaneously by taking the predicted values to predict the value of the next corresponding variable, and so on. *DPS* is applied by implementing the algorithm on both the source and the destination node in a WSN. It uses the predicted values from the models to answer queries, without any communication with the sensor node. The source node continue sensing the data and predicting the values without sending them over the network. The sensed value is sent to the destination only when there is a difference between this value and the predicted value, that is greater than a certain threshold. Indeed, the stronger the correlation among variables, the higher the accuracy of the predicted values.

## 4 Simulation Setup

In order to evaluate our proposition and compare it to existing approaches, we use real sensor values from NREL National Wind Technology Center [21]. We consider the temperature, humidity, global horizontal irradiance[1] and current black photon[2] between 01/06/2019 and 31/07/2019 with a fixed sampling rate of one sample every minute. For each variable, we consider five different thresholds. We note that these thresholds were chosen randomly and can be adjusted for specific QoS needs.

Our approach is compared to *LMS_MOD* [16] and *SARIMA* [13]. These algorithms have different parameters that must be computed. For that, we applied the same methodologies used in the corresponding papers for the tuning process. We would like to point out that the tuning process is repeated five times for each variable based on the different thresholds.

We consider a one hop communication environment with no loss in order to prove the efficiency of our proposal in an optimal case scenario. Simulations were conducted using Python language, and executed in a Raspbian environment, installed on a Raspberry Pi 3. It has a 1.4GHz Arm Cortex-A53 quad-core CPU and 1GB of RAM. The main idea of using Raspberry Pi for simulation is to be able to measure the power consumed by the algorithm (the process of measurement will be detailed in section 5.3).

### *4.1 Data Preparation*

Our proposed solution is based on linear regression. This method uses least squares method in order to calculate its best parameters. However, this method is very sensitive to outliers [22]. For that, before running our algorithm, we cleaned up our dataset by eliminating abnormal observations (i.e., negative values, erroneous values). We note that the dataset is then splitted between training and testing (75% and 25% respectively).

### *4.2 Building Models*

Our algorithm presented in Section 3 is executed taking as input the pre-processed dataset. Table 1 shows the correlation coefficients for the existing variables in the dataset using Pearson correlation coefficient. Based on the values of this matrix, the algorithm creates a prediction model for each variable. Table 2 shows the output of the algorithm. Each row of this table represents a prediction model for a specific variable. The first model is a time series prediction model for temperature. The

[1] Global horizontal irradiance represents the total solar radiation incident on a horizontal surface

[2] Current black photon represents the current generated by photo-voltaic cells

second one uses an *SLR* model, and takes as input the temperature and predicts the humidity. the third and fourth models use the *MLR*, and take as inputs and output the variables denoted by "Input" and "Output" simultaneously in the table.

Table 2: Models structure table showing input/s and output for each model created

| | **Temperature** | **Humidity** | **Irradiance** | **Current BP** |
|---|---|---|---|---|
| **Prediction Model 1** | Output | - | - | - |
| **Prediction Model 2** | Input | Output | - | - |
| **Prediction Model 3** | Input | Input | Output | - |
| **Prediction Model 4** | Input | Input | Input | Output |

## 5 Performance Evaluation

The performance of our algorithm is evaluated using different metrics: Data reduction percentage, *RMSE* and energy consumption. Each of these will be discussed in the following subsections. We note that the temperature variable is not mentioned below in the metrics, since it is predicted in *RADAR* using *SARIMA* model as already detailed in Section 3 (It is the first variable selected in *RADAR* steps).

### *5.1 Data Reduction Percentage*

Data reduction percentage corresponds to the number of packets whose predicted values fall within the range of the chosen threshold, and therefore not transmitted to the destination node. Figures $2 \rightarrow 4$ show the data reduction percentage achieved for our approach (*RADAR*), *LMS_MOD* and *SARIMA*. We can see that *RADAR* presents higher reduction percentage than *LMS_MOD* for the humidity and current black photon. Between 5 and 14% for humidity and between 23 and 34% for current black photon. Concerning the global horizontal irradiance, the data reduction percentage is close between *RADAR* and *LMS_MOD* with a slight improvement for our proposal. This is mainly due to the parameters chosen for each variable in *LMS_MOD*, and more precisely the size of the filter[3] that may impact the data reduction percentage when the filter size is bigger. However, the percentage of data reduction obtained with *SARIMA* is higher than our proposition, and that because its forecasts are based on previous values, that make the predicted values close to the real ones. While in *RADAR*, the prediction is based on regression models which may affect the data

[3] It indicates the number of packets to be transmitted to the destination node, when the model is in the training phase or outdated

reduction percentage in some cases. However, the gain achieved in *SARIMA* in terms of data reduction percentage comes at the cost of a higher error rate and a more consequent energy consumption percentage, which we will detail in the following subsections.

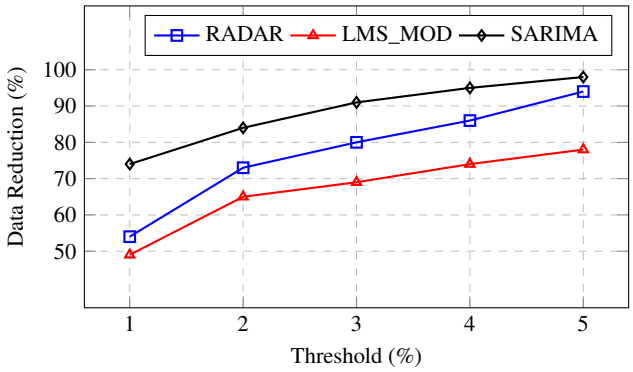


Fig. 2: Data reduction for Humidity

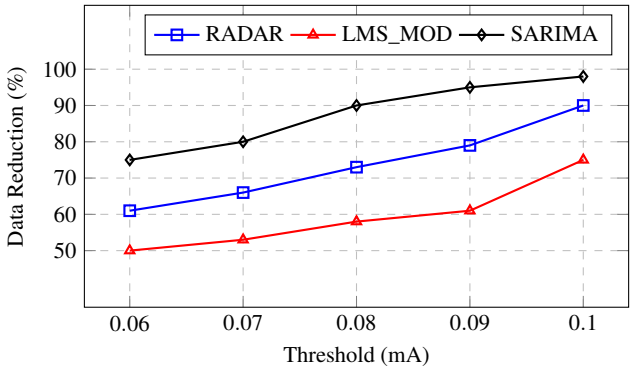


Fig. 3: Data reduction for Current Black Photon

## *5.2 Root Mean Square Error*

In order to identify how close the predicted results are from the real ones, we compute the *RMSE*. It is calculated using the following formula:

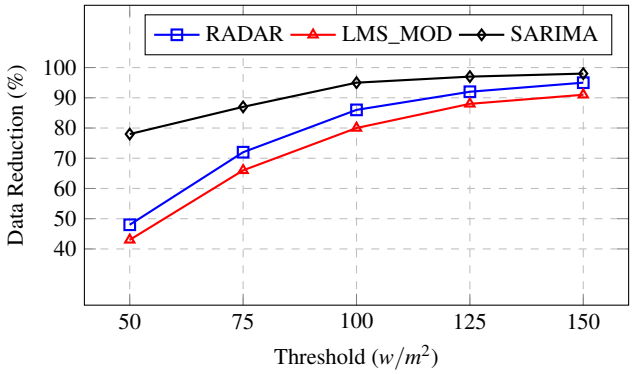


Fig. 4: Data reduction for Global Horizontal Irradiance

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y[n]-u[n])^2} \tag{1}$$

Where $n$ is the number of samples, $u$ and $y$ are two vectors representing the set of real and predicted values respectively.
Figures 5 $\rightarrow$ 7 show the *RMSE* for the three methods, and for the different variables (temperature, irradiance and current black photon). We observe that for the humidity, global horizontal irradiance and current black photon, our proposition has a lower *RMSE* than *LMS_MOD* and *SARIMA*. This is mainly due to the cumulative error that appear in time series models, because of the dependency of a predicted value from the past ones. However, in our model, each value is independent from others, and a prediction error does not affect the upcoming predictions.

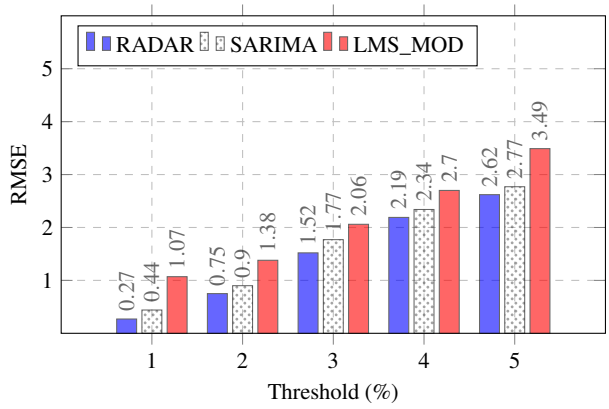


Fig. 5: RMSE for Humidity

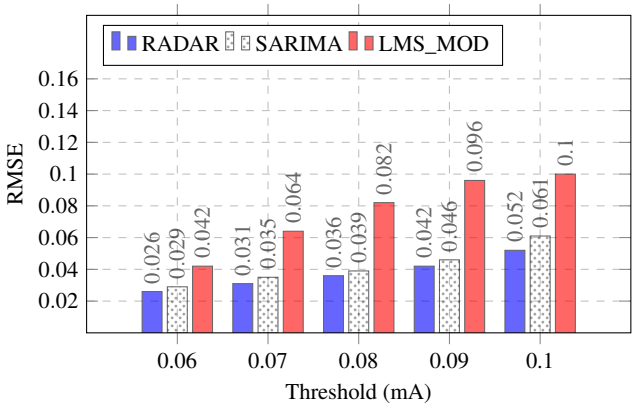


Fig. 6: RMSE for Current BlackPhoton

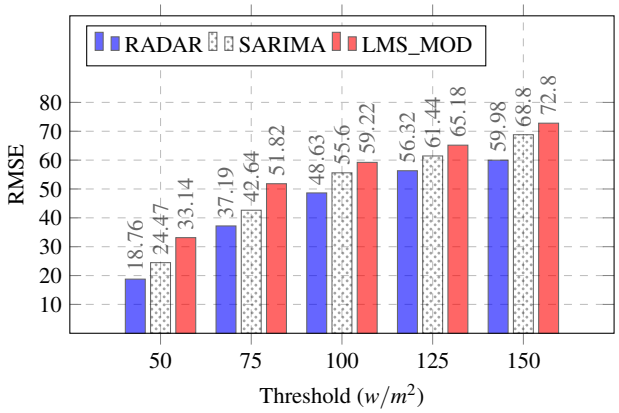


Fig. 7: RMSE for Current Global Horizontal Irradiance

## *5.3 Energy Consumption*

In order to estimate the energy consumed by the Raspberry Pi to execute each algorithm, we used a USB power and energy meter module. It measures the energy consumption during a specific period of time. It is connected to the Raspberry Pi via the USB port.

First, we measure the energy consumption for a period of five minutes, where no tasks are carried out by the Raspberry Pi. Afterwards, the same test for five min-

utes is repeated on the Raspberry for the different algorithms respectively (*RADAR*, *SARIMA* and *LMS_MOD*). The energy consumed by the algorithm is computed using the formula below:

$$E\_Consumed = E\_Algorithm - E\_Idle \quad (2)$$

Where *E_Consumed* is the net energy while executing the algorithm, *E_Algorithm* and *E_Idle* represent the energy consumed by the Raspberry Pi when it's executing an algorithm and when it's not, respectively.

Table 3 shows the energy consumption and the processing time needed by our algorithm (*RADAR*), *LMS_MOD*, and *SARIMA*, to predict the set of variables. We notice that *RADAR* consumes less energy than *LMS_MOD* (10.8 Joules for *RADAR* vs 43.2 Joules for *LMS_MOD*), this is because *LMS_MOD* includes the previous values in the prediction phase, and thereby increases the time required for the prediction task which will consume more energy. Moreover, *RADAR* presents a huge improvement in terms of energy reduction compared to *SARIMA* (10.8 Joules for *RADAR* vs 579.6 Joules for *SARIMA*). This is mainly due to *SARIMA* training and testing phase that require recreating instances of the model so often which will result in a high energy consumption [13]. We would like to point out that in order to predict a set of variables, each one of these requires an instance of a prediction model. Therefore, to predict *X* variables based on *SARIMA* [14] or *LMS_MOD* [16] approaches, *X* time series instances of these models must be created. In *RADAR*, the prediction of *X* variables requires the use of only one instance of the time series model. In other words, in this example, *SARIMA* and *LMS_MOD* use 4 instances each, while *RADAR* uses only one instance of *SARIMA*, thus reducing the energy consumption significantly. The processing time is also more advantageous for *RADAR* compared to *SARIMA* and *LMS_MOD* for the same reasons stated above.

Table 3: Energy consumption and processing time

| | **RADAR** | **LMS_MOD** | **SARIMA** |
|---|---|---|---|
| **Energy Consumption (Joules)** | 10.8 | 43.2 | 579.6 |
| **Processing Time (Milliseconds)** | 27.1 | 29.8 | 153790 |

## 6 Discussion

Before coming to our conclusions, we discuss some relevant issues in our approach. Despite having shown efficiency for different variables, by reducing *RMSE*, energy consumption and ensuring a high reduction percentage, some minor potential issues should be highlighted. Knowing that our approach is based on linear regression models, strong relationships/correlations between variables are required to obtain

accurate predictions. For that, low relationships can affect negatively the performance of our algorithm. In this case, a correlation threshold should then be fixed in order to apply our algorithm. Furthermore, our simulations were conducted on Python considering a no loss scenario of packets over the network. In a real sensor network our model could rise a reliability issue; in a real WSN with interference and losses, if the message containing the reading message transmitted by the sensor is lost, the model at the destination will go apart and the algorithm will predict erroneous values. This should be carefully handled by sending regular control messages per example in order to maintain the synchronization between the sink and the sensor nodes. This issue was not addressed in this work since its main focus was to test the performance of the algorithm in the best cast scenario.

## 7 Conclusion and future work

In this paper, we presented a data prediction correlation based approach, that automatically generates prediction models for different heterogeneous variables. The main advantage of our approach is the ability to adapt to different applications with different requirements as per a SG environment while being energy efficient (if the correlation requirements are satisfied). We tested our approach with real data traces for photo-voltaic cells and performed simulations considering one hop communication networks. *RADAR* provides satisfactory results compared to primary literature solutions. It shows a better performance compared to *LMS_MOD* in terms of *RMSE*, data reduction percentage and energy consumption. Compared to *SARIMA*, it offers a lower error percentage, while consuming much less energy. As future work, we will continue testing our approach on different applications. Later on, we will implement our algorithm in a real WSN scenario to evaluate its performance.

## Acknowledgments

This work was partially funded by a grant from the SoMel SoConnected project. This project involves the MEL (Métropole Européenne de Lille), Enedis, EDF, Dalkia, Intent, the Lille Economie-Management Laboratory, HEI - Yncréa HdF and the Faculties of the Catholic University of Lille.

## References

1. J. Nassar, “Ubiquitous networks for smart grids,” Ph.D. dissertation, Universite des Sciences et Technologies de Lille, 2018.
2. D. Ilic, P. G. Da Silva, S. Karnouskos, and M. Griesemer, “An energy market for trading electricity in smart grid neighbourhoods,” in *6th IEEE international conference on digital*

*ecosystems and technologies (DEST)*. IEEE, 2012, pp. 1–6.
3. U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "What does model-driven data acquisition really achieve in wireless sensor networks?" in *IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2012, pp. 85–94.
4. G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad hoc networks*, vol. 7, no. 3, pp. 537–568, 2009.
5. J. Nassar, M. Berthomé, J. Dubrulle, N. Gouvy, N. Mitton, and B. Quoitin, "Multiple instances qos routing in rpl: Application to smart grids," *Sensors*, vol. 18, no. 8, p. 2472, 2018.
6. B. Chreim, J. Nassar, and C. Habib, "Regression-based data reduction algorithm for smart grids," in *IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, 2021.
7. N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, vol. 2. IEEE, 2005, pp. 8–13.
8. M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap, "Aggregation functions: means," *Information Sciences*, vol. 181, no. 1, pp. 1–22, 2011.
9. A. Skordylis, A. Guitton, and N. Trigoni, "Correlation-based data dissemination in traffic monitoring sensor networks," in *Proceedings of the 2006 ACM CoNEXT conference*, 2006, pp. 1–2.
10. T. B. Matos, A. Brayner, and J. E. B. Maia, "Towards in-network data prediction in wireless sensor networks," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 592–596.
11. J. M. Bahi, A. Makhoul, and M. Medlej, "An optimized in-network aggregation scheme for data collection in periodic sensor networks," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2012, pp. 153–166.
12. G. M. Dias, B. Bellalta, and S. Oechsner, "The impact of dual prediction schemes on the reduction of the number of transmissions in sensor networks," *Computer Communications*, vol. 112, pp. 58–72, 2017.
13. C. Liu, K. Wu, and M. Tsao, "Energy efficient information collection with the arima model in wireless sensor networks," in *GLOBECOM'05. IEEE Global Telecommunications Conference*, vol. 5. IEEE, 2005, pp. 5–pp.
14. S. V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal arima model with limited input data," *European Transport Research Review*, vol. 7, no. 3, p. 21, 2015.
15. S. Santini and K. Romer, "An adaptive strategy for quality-based data reduction in wireless sensor networks," in *Proceedings of the 3rd international conference on networked sensing systems (INSS 2006)*. TRF Chicago, IL, 2006, pp. 29–36.
16. J. Nassar, K. Miranda, N. Gouvy, and N. Mitton, "Heterogeneous data reduction in wsn: Application to smart grids," in *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*, 2018, pp. 1–6.
17. C. G. N. de Carvalho, D. G. Gomes, J. N. de Souza, and N. Agoulmine, "Multiple linear regression to improve prediction accuracy in wsn data reduction," in *7th Latin American Network Operations and Management Symposium*. IEEE, 2011, pp. 1–8.
18. N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, "Predicting solar generation from weather forecasts using machine learning," in *IEE international conference on smart grid communications (SmartGridComm)*. IEEE, 2011, pp. 528–533.
19. S. Kumar, "Solar energy prediction using machine learning," Tech. Rep., 2015.
20. K. H. Zou, K. Tuncali, and S. G. Silverman, "Correlation and simple linear regression," *Radiology*, vol. 227, no. 3, pp. 617–628, 2003.
21. D. Jager and A. Andreas, "Nrel national wind technology center (nwtc): M2 tower; boulder, colorado (data)," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep. DA-5500-56489, 1996. [Online]. Available: http://dx.doi.org/10.5439/1052222
22. T. M. Hope, "Linear regression," in *Machine Learning*. Elsevier, 2020, pp. 67–81.