# Machine Learning Based Data Reduction in WSN for Smart Agriculture

Christian Salim, Nathalie Mitton

# Machine Learning Based Data Reduction in WSN for Smart Agriculture

Christian Salim and Nathalie Mitton

**Abstract** Nowadays, the agriculture domain faces a lot of challenges for a better usage of its natural resources. For this purpose, and for the increasing danger of climate change, there is a need to locally monitor meteorological data and soil conditions to help make quicker and more adapted decision for the culture. Wireless Sensor Networks (WSN) can serve as a monitoring system for those types of features. However, WSN suffer from the limited energy resources of the motes which shorten the lifetime of the overall network. Every mote periodically captures the monitored feature and sends the data to the sink for further analysis depending on a certain sampling rate. This process of sending big amount of data causes a high energy consumption of the sensor node and an important bandwidth usage on the network. In this paper, a Machine Learning based Data Reduction Algorithm (MLDR) is introduced. MLDR focuses on environmental data for the benefits of agriculture. MLDR is a data reduction approach which reduces the amount of transmitted data to the sink by adding some machine learning techniques at the sensor node level by keeping data availability and accuracy at the sink. This data reduction helps reduce the energy consumption and the bandwidth usage and it enhances the use of the medium while maintaining the accuracy of the information. This approach is validated through simulations on MATLAB using real temperature data-sets from Weather-Underground sensor network. Results show that the amount of sent data is reduced by more than 70% while maintaining a very good accuracy with a variance that did not surpass 2 degrees.

## 1 Introduction

New and improved methods are needed for modern agricultural fields. Some additional challenges appear with the alarming climate change and scarcity of water

Christian Salim and Nathalie Mitton
Inria, France, e-mail: firstname.lastname@inria.fr

demand. Consequently, the need for automation and intelligent decision making is becoming more important [9]. Wireless sensor networks (WSN) serve as low-cost monitoring systems in different domains (healthcare, industrial, video surveillance, environmental, etc...) [3]. We assume a WSN deployed for smart agriculture, periodically gathering environmental data from different sensor nodes and sends the data to a sink for further analysis [8].

WSNs must take account of the medium occupancy of the network, the bandwidth usage and most importantly the limited energy resources of the sensor nodes. This periodic cycle leads to a lot of redundant data sent to the sink, especially if no changes occur in the monitored feature (e.g. if the temperature stays stable). This big amount of periodic data is even more challenging at the sensor node level. For instance, for the temperature, each node captures and sends a value every 30 min, 48 values per day. And this being performed by hundreds of nodes and not only for temperature but also for other values, potentially more frequent than the temperature. To face those challenges, and to reduce the amount of data sent from the sensor nodes, we propose Machine Learning based Data Reduction Algorithm (MLDR), a data reduction technique based on a machine learning process to predict on a dual prediction basis [7], the next values of the monitored feature (e.g: temperature) both at the sensor node and at the sink simultaneously [15]. As such, MLDR reduces the energy consumption of the communication process, since the node does not need to send all the captured values to the sink anymore while still ensuring high data accuracy at the sink. In MLDR, data is sent if and only if the real value is too different of the predicted one, even in the learning phase. The status is always updated at the sink. The prediction process starts at the end of the learning phase at both the sensor node and the sink simultaneously. Results show the efficiency of our approach that maintains a high accuracy at the sink while sending less than 70% of the data.

The remainder of this paper is as follows: Section 2 introduces the state of the art, Section 3 presents our model and the MLDR algorithm with its several instances. Different case studies are presented in Section 4. In Section 5, some experimental results validate the approach. Finally, Section 6 concludes the paper.

## 2 Related Work

Data reduction for WSN has already received much attention these last years. Main data reduction techniques that are traditionally used can be classified as clustering, scheduling, compressive sensing, multi-paths multi-channels, data correlation and machine learning (dual prediction model). In this section we will list and explain some of these approaches.

Clustering methods aim to reduce the energy consumption in WSN by detecting spatio-temporal data similarities [16],[18] and [11]. In [11], the authors propose a method to re-cluster the nodes using a fuzzy inference system. Several parameters are taken into consideration to achieve better energy saving such as: the average sensed data rate of cluster members, the distance at which the member nodes are

from the sink and the power of cluster head nodes. Machine learning is applied to classify data based on their similarity on every cluster member node for less transmission.

Other papers worked on data correlation and data similarity between several features as in [16],[5],[13],[12],[1] and [4]. Authors in [12] present a Bayesian Inference Approach to detect data with high spatio-temporal correlated data, to avoid transmitting data that can be reconstructed from another data such as temperature and humidity in some cases. This interesting approach requires the availability of different correlated data. Our approach applies to a single data monitoring and could be completed by such an approach.

Adapting the sampling rate of the sensor nodes can be a solution to reduce the amount of sent data from the sensor nodes to the sink using different approaches: machine learning and the history of the data as in [15], [16], [14] and [6], the channel selection using a Thompson sampling based approach as in [17], and the cubic adaptive sampling as the authors proposed in [7]. Our approach focuses on the amount of transmitted data and not on the amount of captured data.

Machine learning for data prediction is widely used for data reduction as in [15],[7],[18],[10], [2],[14] and [6]. In the dual prediction model [15] [7], the sensor node and the sink both predict the next values of the monitored feature simultaneously. In [15], the authors propose a machine learning technique AS+TR to predict the next values, while sending all the data in the learning phase to the sink. In this approach, they take the uniform trend into consideration but do not consider a changeable trend between consecutive values as we do. Also they do detect a trend directly after a single change, which can cause some problems for the learning process and send more values. In this paper, we tackle the challenge of data reduction in wireless sensor networks deployed to monitor the environmental parameters for agricultural surveillance taking account of the limited energy resources of every sensor node. Our solution for data reduction at the sensor node is to implement a light data reduction algorithm to reduce the amount of sent data to the sink. It is based on the dual prediction model in [15] while modifying the learning phase process and the prediction functions in different ways. However to further enhance our approach of data reduction in the future, the clustering, spatio-temporal correlation and adaptive sampling techniques can be adopted to further reduce the amount of data captured, processed and transmitted on the network.

## 3 Machine Learning based Data Reduction algorithm (MLDR)

### *3.1 Overview*

In this section, we present our machine learning based data reduction algorithm (MLDR) and its main principles. In MLDR, while capturing data, the nodes start a learning phase to detect the behavior of the monitored parameter (later referred as

trend in our paper). A trend is detected once the variation of the given parameter over time is stable (e.g temperature passing from 9 to 10 to 11 degrees, it means the trend is 1 degree per 1 time unit). Once a trend is detected, the sensor node sends this trend to the sink with the last value of the parameter from the learning phase. Afterwards, all the real measurements are not sent to the sink unless a critical change is detected depending on predefined thresholds. The sink and the sensor node start a dual prediction mechanism to predict the next values based on the last sent value and the trend. Meanwhile, the sensing process is not affected at the sensor node side. For each new captured value, the sensor node compares the predicted value with the real captured value. If a critical change is noticed (even in the learning phase), the sensor node sends the critical value to the sink as an information that a change happened to stop unnecessary predictions. Then, the node enters the learning phase again, trying to detect a new trend. Two variations have been introduced in our study, the Hold (MLDR-H) and the Buffer (MLDR-B), that differ in the quantity of calculations to be performed on the nodes. Indeed, sensor nodes are memory-limited devices and cannot always offer to store too much history data. So, based on the capacity of nodes, we can use one variation or the other. In the MLDR-Hold variant, the sensor node sends a hold message to the sink to stop the predictions and goes back to the learning phase trying to detect a new trend from the new values.
In the MLDR-Buffer variant, the mote uses a buffer to store the last values, when a change occurs, it goes back to the buffer to compute a new trend directly (if possible). This trend, if found, is directly sent to the sink to continue the accurate predictions. If the trend is not found, the mote starts the learning phase again.

### *3.2 Machine Learning based Data Reduction Algorithm*

In this section, we present in details both variations of the machine learning based data reduction algorithm (MLDR) in Algorithms 1 and 2.

In both variations, the sensor node sends the first captured value $val_0$ to the sink. Then the sensor node enters a learning phase for $n$ consecutive values to detect the changing trend $tr$ between those values.

To compute the trend, the sensor node must compare the differences between $n$ consecutive values as follows:

$$\begin{aligned} tr_1 &= val_1 - val_0 \\ tr_2 &= val_2 - val_1 \\ ... &= ..... - ..... \\ tr_i &= val_{i+1} - val_i \end{aligned} \tag{1}$$

$$\begin{aligned} d_1 &= |tr_1 - tr_0| \\ d_2 &= |tr_2 - tr_1| \\ \ldots &= \ldots. - \ldots. \\ d_i &= |tr_{i+1} - tr_i| \end{aligned} \tag{2}$$

where vector $tr\{tr_0; tr_1; \ldots; tr_i\}$ represents the trends between values (the change of the values with time) and vector $d\{d_0; d_1; \ldots; d_i\}$ represents the differences between those trends.

## *3.3 Trend Behavior*

The trend behavior is used to predict the next values. It is computed in parallel at the node and to the sink on the basis of the values sent by the sensor node.

We assume that predicting a weather value based on its history is like predicting a trajectory. We thus adopt the kinematics functions for this type of prediction replacing the movement, speed and acceleration by value (val), trend (tr) and evolution (e) respectively. In the Equation 3, the trajectory function represents the new values of the feature in time:

$$val = \frac{1}{2} e \times t^2 + tr_0 \times t + val_0 \tag{3}$$

The trend is represented by the speed function which is the first derivative of the trajectory function.

$$tr = e \times t + tr_0 \tag{4}$$

The evolution $e$ of the trend is represented by the acceleration which is the first derivative of the speed and the second derivative of the trajectory. It can be stable (evolution $e = 0$) or unstable ($e = constant$ or unstable $e$). Different cases can occur: stable trend, unstable trend with stable or unstable evolution.

### 3.3.1 Stable Trend

If the trend is stable, it means the evolution is null $e = 0$ or that the differences between consecutive trends are negligible and do not surpass a certain threshold. In this case, the node will send the $n^{th}$ value with the last trend computed to the sink. The stability is detected when Equation 5 holds.

$$d_i < th_{tr} \quad \forall \quad 0 < i <= n \tag{5}$$

where $th_{tr}$ is the threshold to define stability (it can be learnt through several days and/or multiple observations or predefined by specialists).

Once the trend is sent to the sink with the latest value, both the sensor and the sink can predict the next values using the kinematic functions defined in 3 and 4

with $e = 0$ and $t = 1$ for every period. In the case of an absurd change, the new value is sent directly to the sink even if the node is in the learning process. The difference between the captured value and the predicted value $pval_i$ at a period $i$ must not surpass this threshold (Equation 6).

$$|val_i - pval_i| < th_{cr} \tag{6}$$

#### 3.3.2 Unstable Trend

A trend is called unstable when several differences between consecutive values overpass a certain threshold. This evolution could be constant or not.
**Stable Evolution:** If the values of the feature are uniformly accelerated or decelerated, the evolution is constant but not null. In this case, in the learning phase, if the trend is not stable but its evolution is stable for $n$ consecutive samples, the mote sends the last value, the trend and its evolution to the sink. Once the sink receives the data, the prediction process starts. The trajectory function represents the new values of the temperature with time based on the trend and evolution for each period ($t = 1$) as shown in Eq. 7.

$$\Delta val = \frac{1}{2}e + tr_0 \tag{7}$$

The trend is represented by the speed function which is the first derivative of the trajectory function and for $t = 1$ as set in Eq. 8.

$$\Delta tr = e \tag{8}$$

**Unstable Evolution:** If the values of the feature are accelerated or decelerated in a non uniform way, the evolution $\Delta e$ is unstable. In this case, the sensor node remains in the learning phase until one of the above situations is detected (stable trend or unstable trend with stable evolution). Beside staying in the learning phase and taking off the prediction process, the sensor node will only send the values that exceed a certain threshold as shown in Equation 6.

## 4 Case Study

In this section, we are interested to study both variants of MLDR algorithm once a change occurs. Our case study is done on the stable trend scenario between values.

### *4.1 Hold Scenario*

We consider that a trend is computed and the dual prediction process is taking place. In the Hold scenario, when a change occurs, the sensor node sends to the sink a hold

message (to stop useless predictions) and the value where the change happened. The sensor then computes a new trend based on several consecutive captured values as shown in Figure 1 and Algo. 1. In this scenario, we may lose some updates if the sensor needs time to detect the new trend or if a new trend does not exist. However, if a big change occurs, it will be directly detected and sent to the sink if it surpasses a certain predefined threshold even in the learning phase. Once a hold message is sent to the sink, the latter keeps the last value as a valid one, until a new value is received. The accuracy of the data is poorly affected since the node sends again a new value if a drastic change happened (a critical threshold $th_{cr}$ is always present).

**Algorithm 1** Machine Learning Data Reduction Hold Algorithm MLDR-H

Send the first Value $val_0$
Set $val_{cmp}$=$val_0$, $th_{cr}$, $th_{tr}$, $i = 1$, $tr = 0$, $e = 0$ # Enter the Learning Phase
**while** $Energy > 0$ **do**
  **for** each period $j$ **do**
    **if** $\|val_i - val_{cmp}\| > th_{cr}$ **then**
      Send $val_i$ ; Set $val_{cmp}$=$val_i$
    **end if**
    Compute $d_i$, $tr_i$, $e_i$; Set $i = i + 1$
    **if** $d_i > th_{tr}$ **then** Restart the Learning phase **end if**
  **end for**
  Send $val_i$, tr=$tr_i$, e=$e_i$ # Start the Prediction Process
  Set $pval_i$=$val_i$ and $val_{cmp}$=$val_i$
  **for** each period $i$ **do**
    Compute $pval_i$ # using Equ. 3 and 4
    Set $val_{cmp}$=$pval_i$
    **if** $\|val_i - val_{cmp}\| > th_{cr}$ **then**
      Send $val_i$ and $HOLD$ message
      Go to the Learning phase
    **end if**
  **end for**
**end while**

## *4.2 Continuous Trend*

As in the last section, we consider that a trend is computed and the dual prediction process is taking place at the sensor node and at the sink levels to predict the next values. In the Buffer scenario, we consider that every sensor node has a memory to stock the $n$ values needed to re-compute the trend (like a sliding window). Once a change is detected, the new trend can be re-computed using the inner buffer of the sensor node. Thus, the sensor node sends the new trend, its evolution and the last value directly to the sink. In this case, there is no useless predictions and the new ones are based on the new trend as shown in Figure 2 and algorithm 2. If a trend is not detected, the node goes back to the learning phase trying to find a new

trend from the new values. In this case, the sink loses some accuracy but, as we said before, the difference is limited because of the critical threshold.

**Algorithm 2** Machine Learning Data Reduction Buffer Algorithm MLDR-B

```
    Set th_cr, th_tr, n, i = 1, tr = 0, Buffer[n]
 2: Run MLDR-H Learning Phase                    # Start the Prediction Process
    Set pval_i=val_i and val_cmp=val_i
 4: for each period i do
      Update Buffer[n] by val_i
 6:   if ||val_i − val_cmp|| > th_cr then
        for each value k: 1 to n in Buffer[n] do
 8:       Compute tr_k and d_k
          if d_k > th_tr then Go to the Learning phase end if
10:     end for
        if k = n then Set tr = tr_k end if
12:     Send val_i and tr
      end if
14: end for
```

## *4.3 Behavior Comparison*

### 4.3.1 MLDR-H

Figure 1 shows the MLDR-H behavior. In this figure, the minimal number of values to detect a trend is $n = 4$. Once the trend is detected, the sensor sends it to the sink with the last captured value. The prediction process starts at the node and at the sink levels simultaneously. As shown in Figure 1, the sensor node re-enters a learning phase once a change occurs. While in the learning phase, the node sends a new critical value (surpassing the threshold). The second learning phase needed 7 to 8 values to detect the trend because of non stability in the captured values.

### 4.3.2 MLDR-B

Figure 2 shows the MLDR-B behavior. It is similar to MLDR-H except a modification about the trend modification. Here, the last $n$ values are always buffered and updated. If a change occurs, the node can immediately compute the new trend from the observations conserved in the buffer. If the trend or the evolution is stable, the process works and is beneficial. If it does not work and the node does not define a new trend from the buffer, the original learning phase takes place again.

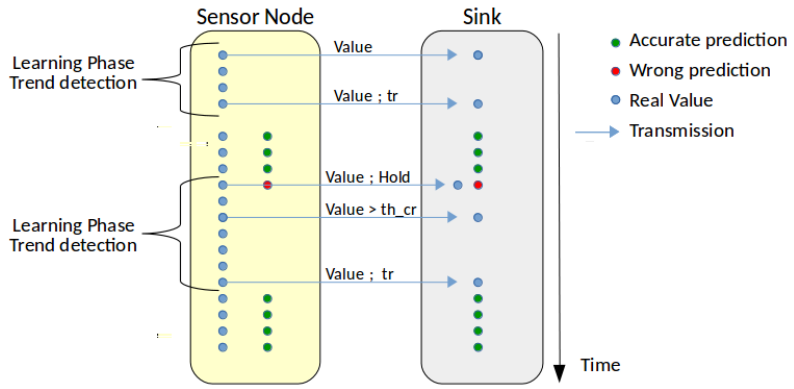


**Fig. 1** Sensor node behavior using MLDR-H algorithm

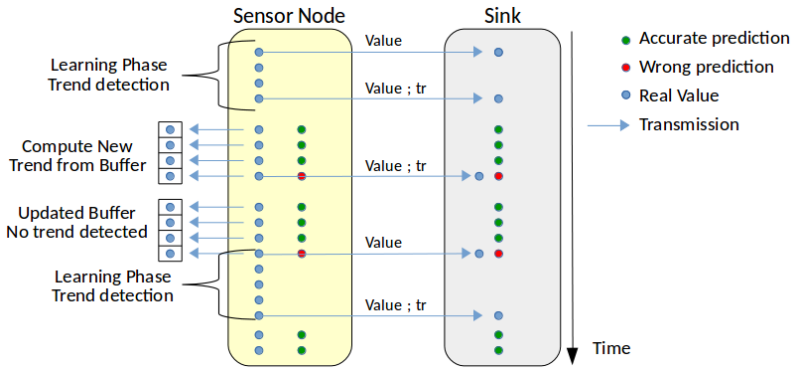


**Fig. 2** Sensor node behavior using MLDR-B algorithm with buffer size $n = 3$

## 5 Experimental Results

In this section, we compare MLDR to AS_TR from [15]. We used a MATLAB simulator with a temperature dataset for the $27^{th}$, $28^{th}$, $29^{th}$ and the $30^{th}$ of October 2019 in Lille city, France, from Weather_Underground website which gathers data from a sensor network of different weather stations deployed around the globe[1]. The sampling rate of the sensor node is set to 1 value each 30min. The temperature in those 4 days varied from 17 degrees Celsius as a maximum to 2 degrees Celsius as a minimum. We applied the two versions of MLDR algorithm. In those experiments, the thresholds are set as $th_{tr}$=1 degree and $th_{cr}$=2 degrees.

192 samples are measured at the node. The main purpose is to reduce this amount of data sent from the sink while maintaining the information integrity. While apply-

[1] https://www.wunderground.com

ing our algorithms MLDR-H and MLDR-B, the number of sent values is decreased from 192 to 25 and 27 respectively. Due to the direct computation of the new trend when a change occurs, the integrity of the values is slightly more preserved using MLDR-B, however it can be less energy-efficient since it stores the values and computes the trend continuously. Comparing those methods with the approach proposed in [15], each node in MLDR sends only the critical data when it occurs, waits until the end of the learning phase to sent the reference value and the trend, but not all the captured data in the learning phase. In Figure 3 and Table 1, we draw a comparison

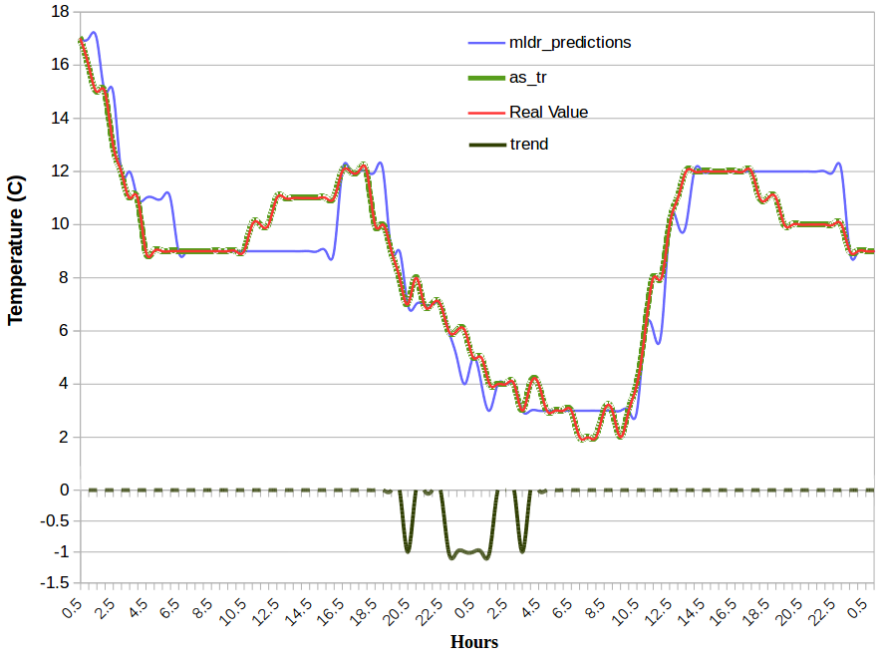


**Fig. 3** MLDR Predictions Behavior

between our data reduction approach and the data reduction part in [15] using the same parameters.

The number of predicted values and their integrity for the first two days are shown in Figure 3. The maximum difference between the predicted values in MLDR and the real values varies between 0 and 2 degrees Celsius. The trend was 0 for several periods because of the stability of the temperature in different parts of the day. In other periods, there were no stability in the change so the trend could not be detected, the learning phase must run again to detect a stability of the change.

AS+TR approach [15] has a high level of data prediction accuracy, however, as shown in Table 1, it sends three times more data than MLDR which still maintains a high (Figure 3) (less than 2 degrees of difference). In [15], the node enters the

**Table 1** Amount of transmitted data per day

| Day | All data | MLDR-H | MLDR-B | AS+TR[15] |
|---|---|---|---|---|
| $27^{th}$ | 48 | 8 | 9 | 23 |
| $28^{th}$ | 48 | 10 | 11 | 29 |
| $29^{th}$ | 48 | 5 | 5 | 17 |
| $30^{th}$ | 48 | 2 | 2 | 16 |
| Total | 192 | 25 | 27 | 85 |
| Data Reduction | 0% | 87% | 85% | 56% |

learning phase upon any change and the trend is computed after only 1 difference. This process leads to get directly real data from the sensor node because of the difference between the real value and the predicted value based on the changeable trend, which explains the big amount of sent data from the node to the sink as shown in Table 1. The main difference between MLDR-H and MLDR-B is not based on the amount of data reduction but on data accuracy. The MLDR-B, if possible, sends a new trend to continue the accurate prediction process. Meanwhile, in MLDR-H, we must wait at least one learning phase to detect the new trend which can lead to some differences between the real temperature and the temperature at the sink.

## 6 Conclusion and Future Work

In this paper, we proposed a data reduction technique based on a simple machine learning technique in WSN implemented for agriculture to detect any abnormal situation in the meteorological data which can harm the agriculture. Our simulations show a reduction of more than 70% of the overall data and 30 to 50 % while comparing with other methods.

In the near future, several additional ideas can be added to our data reduction technique to help face the problems generated by the big data flow of wireless sensor nodes, to have a more complete work such as data correlation (e.g analysis of the correlation between temperature, humidity and wind speed)[13] in space and time, or adaptive Sampling Rate (in stable mode the sensors can capture 1 value each 30min or an hour). For example in all the cases where the trend is 0 for consecutive periods, a scheduling technique can be adopted to schedule the sensing process, or the sampling rate of the nodes is adapted for the same purpose.

## Acknowledgments

This work was partially supported by a grant from CPER DATA and by LIRIMA Agrinet project.

## References


1. Azaza, M., Tanougast, C., Fabrizio, E., Mami, A.: Smart greenhouse fuzzy logic based control system enhanced with wireless data monitoring. ISA Transactions **61**, 297 – 307 (2016)
2. Balducci, F., Impedovo, D., Pirlo, G.: Machine learning applications on agricultural datasets for smart farm enhancement. Machines **6**(3) (2018)
3. Díaz, S.E., Pérez, J.C., Mateos, A.C., Marinescu, M.C., Guerra, B.B.: A novel methodology for the monitoring of the agricultural production process based on wireless sensor networks. Computers and Electronics in Agriculture **76**(2), 252 – 265 (2011)
4. Ghaddar, A., Razafindralambo, T., Simplot-Ryl, I., Tawbi, S., Hijazi, A.: Algorithm for data similarity measurements to reduce data redundancy in wireless sensor networks. In: Inter. Symp. on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) (June 2010)
5. Ghaddar, A., Razafindralambo, T., Simplot-Ryl, I., Simplot-Ryl, D., Tawbi, S., Hijazi, A.: Investigating Data Similarity and Estimation Through Spatio-Temporal Correlation to Enhance Energy Efficiency in WSNs. Ad Hoc & Sensor Wireless Networks **16**(4), 273–295 (Dec 2012)
6. Habib, C., Makhoul, A., Darazi, R., Salim, C.: Self-adaptive data collection and fusion for health monitoring based on body sensor networks. IEEE Transactions on Industrial Informatics **12**(6), 2342–2352 (Dec 2016)
7. Monteiro, L.C., Delicato, F.C., Pirmez, L., Pires, P.F., Miceli, C.: Dpcas: Data prediction with cubic adaptive sampling for wireless sensor networks. In: International Conference on Green, Pervasive, and Cloud Computing. pp. 353–368. Springer (2017)
8. Musaazi, K.P., Bulega, T., Lubega, S.M.: Energy efficient data caching in wireless sensor networks: A case of precision agriculture. In: Nungu, A., Pehrson, B., Sansa-Otim, J. (eds.) e-Infrastructure and e-Services for Developing Countries (2015)
9. Ojha, T., Misra, S., Raghuwanshi, N.S.: Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges. Computers and Electronics in Agriculture **118**, 66 – 84 (2015)
10. Patil, S.S., Thorat, S.A.: Early detection of grapes diseases using machine learning and iot. In: Inter. Conf. on Cognitive Computing and Information Processing (CCIP) (Aug 2016)
11. Radhika, S., Rangarajan, P.: On improving the lifespan of wireless sensor networks with fuzzy based clustering and machine learning based data reduction. Applied Soft Computing **83** (2019)
12. Razafimandimby, C., Loscri, V., Vegni, A.M., Neri, A.: Efficient bayesian communication approach for smart agriculture applications. In: IEEE Vehicular Technology Conf. (VTC-Fall) (Sep 2017)
13. Razafimandimby, C., Loscri, V., Maria Vegni, A., Aourir, D., Neri, A.: A Bayesian approach for an efficient data reduction in IoT. In: Int. Conf. on Interoperability in IoT (InterIoT) (Nov 2017)
14. Salim, C., Makhoul, A., Darazi, R., Couturier, R.: Similarity based image selection with frame rate adaptation and local event detection in wireless video sensor networks. Multimedia Tools and Applications **78**(5), 5941–5967 (Mar 2019)
15. Tayeh, G.B., Makhoul, A., Laiymani, D., Demerjian, J.: A distributed real-time data prediction and adaptive sensing approach for wireless sensor networks. Pervasive and Mobile Computing **49**, 62 – 75 (2018)
16. Tayeh, G.B., Makhoul, A., Perera, C., Demerjian, J.: A spatial-temporal correlation approach for data reduction in cluster-based sensor networks (2019)
17. Toldov, V., Clavier, L., Loscrí, V., Mitton, N.: A thompson sampling approach to channel exploration-exploitation problem in multihop cognitive radio networks. In: IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) (Sep 2016)
18. Wu, M., Tan, L., Xiong, N.: Data prediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications. Information Sciences **329**, 800 – 818 (2016), special issue on Discovery Science