Simba: Similar-evolution based Aggregation in Wireless Sensor Networks

Jin Cui, Fabrice Valois

Simba: Similar-evolution Based Aggregation in Wireless Sensor Networks

Jin CUI, Fabrice VALOIS
Université de Lyon, INSA-Lyon
INRIA, CITI, F-69621 Villeurbanne, France
Email: {jin.cui,fabrice.valois}@insa-lyon.fr

***Abstract*—Data aggregation is an important mechanism to reduce energy consumption in Wireless Sensor Networks (WSNs). By investigating spatial and/or temporal correlation of raw data, sensor nodes can aggregate raw data to a meaningful digest instead of directly sending raw data to sink, this process is considered as data aggregation. Several aggregation works focus on the raw data, they use raw data to cluster the nodes or to do aggregation. While analysis of datasets of real projects shows that some nodes perform similar evolution. Thus we propose a raw data-independent aggregation, i.e., *Sim*ilar-evolution *B*ased *A*ggregation (Simba), to consider the evolution of data rather than the raw data. *Simba* creates a group out of isolated nodes, nodes in the group can cooperatively execute data aggregation, this process reduces the energy consumption on each node. Besides, similar evolution of nodes guarantees the recover accuracy. Our experiments demonstrate that *Simba* can save more than 91% energy comparing no aggregation, and save more 30% energy than original aggregation functions, and *Simba* can recover data with higher fidelity comparing with the works relying on raw data.**

***Keywords*—Data aggregation; temporal-spatial correlation; similar evolution; wireless sensor networks.**

I. Introduction

As a wireless networking solution, wireless sensor networks (WSNs) suffer from problems like network congestion, packet loss. Moreover, sensor nodes have intrinsic limitations, e.g., low computation and energy constraint. All of these problems are challenging the WSNs techniques. To solve these problems, data aggregation has been proposed. Data aggregation is a data gathering strategy with the idea of using data correlation to reduce the amount of data. In general, spatial and/or temporal correlation may exist in the data of WSNs. Spatial correlation occurs in data collected by neighbouring nodes; and data collected by a node at different time instants may lead to temporal correlation. The correlations can help sensor nodes to aggregate raw data as a digest, and nodes only send this digest to sink.

Due to the spatial and/or temporal correlation, most dedicated aggregation works highlight the role of raw data, such as [1], [2]. These works point out that the nodes holding same or close data (close here is defined by a threshold) can be clustered together to reduce traffic, and can achieve the goal of aggregation. These works reveal importance of raw data and the correlation, but we know sensor nodes are error-prone that may lead to raw data have errors and perform dynamically. We find two properties from real datasets: 1) even if the environment is quite similar, the raw data are not necessarily close; 2) although nodes do not have similar raw data, but some nodes have similar evolution. From the first property, we know that the works involving raw data need to pay more attention on choosing appropriate threshold, and they need to conquer the dynamic change of raw data. The second property reveals an interesting phenomena, neighbouring nodes often show the similar evolution even though the raw data are not totally same. The evolution has spatial-temporal features, spatial feature is shown from the location of the node, and temporal feature is shown from the sequence of data.

Therefore, we propose a raw data-independent solution: **Sim**ilar-evolution **B**ased **A**ggregation (Simba). *Simba* takes benefits from groups, and the groups are built by similarity of evolution. In a network, *Simba* introduces two phases: set-up phase and aggregation phase. Set-up phase is group forming phase, all the nodes use a vector ($\vec{Rc}$) to demonstrate evolution, and then by communicating with neighbours, nodes holding similar evolution form a group. A group leader *GL* will be selected to represent the group. Aggregation phase is aggregation function executing phase, when data of *GL* are recovered by sink, the data of other nodes in the same group can be easily computed. We provide aggregation functions, A-ARMA [3], polynomial aggregation [4] and average, to test the performance with and without Simba. The simulation results from Matlab and WSNet [5] show that, *Simba* can save more 30% energy than original functions, and save more than 91% energy comparing with no aggregation. In the meanwhile, sink can recover data with high fidelity according to RMS computation.

The main contributions of our work are listed as:

- We investigate the real datasets, and analyse two key properties: 1) even if the environment is quite similar, the raw data are not necessarily close; 2), although nodes do not have similar raw data, but some nodes have similar evolution.
- We propose *S*imilar-evolution *B*ased aggregation. *Simba* groups several nodes together by similar evolution, and nodes execute aggregation functions by the unit of group. The experiments show that *Simba* reduces energy consumption and guarantees the recover accuracy in the meanwhile.

The rest of paper is organized as follow. Prior data aggregation techniques are reviewed in Section II. We discuss our analysing, and present how to model evolution in Section III. We propose similar evolution-based aggregation protocol in Section IV and evaluate the performance in Section V. Finally, we conclude our work in Section VI.

II. Related works

To the best of our knowledge, there is few works concerned on data evolution, many works pay attention on value of raw data. In [1], the authors propose to draw a map for sensors, and the isoline is defined by difference between raw data. It means nodes in same isoline share same raw data (under some threshold), only when an isoline changes, corresponding nodes need to update new data. [2] propose characteristic correlation to group nodes with same raw data, they highlight that same or similar raw data can provide characteristic correlation between nodes, and which is more precise than spatial distance of nodes. In addition, they provide a "categorizing range" to define the error of similar raw data, smaller range is, similar raw data are. When sink receive the representative of virtual cluster, it retrieves the value as all the cluster members readings. They only focus on raw data, not the evolution of data series, which leads to frequently change cluster or group when threshold is too sensitive or raw data fluctuate.

We also review the aggregation functions we considered. Average is a traditional aggregation function, sensor nodes send average value to sink replacing several raw data. A-ARMA is proposed in [3], which provide to use ARMA model to forecast the data. An ARMA model can be formulated as:

$$\begin{aligned}\mathcal{F}_{arma} \equiv \hat{x}_\tau =& \varphi_0 + \varphi_1 x_{\tau-1} + \cdots + \varphi_p x_{\tau-p} \\ &+ \vartheta_1 \varepsilon_{\tau-1} + \cdots + \vartheta_q \varepsilon_{\tau-q}\end{aligned} \tag{1}$$

where $\hat{x}_\tau$ is predicted value, $\{x_{\tau-p}, \cdots, x_{\tau-1}\}$ are time series, ε is error term, and φ, ϑ are model coefficients. A node builds ARMA model (as Eq. 1), and the coefficients (φ, ϑ) are sent to the sink instead of raw data. After collecting next S data, the node detects the RMS error between predicted data $\hat{x}_\tau$ and the raw data. If the difference is lower than th_{err}, the node continues using its current ARMA model. Otherwise, the node computes the new coefficients on the latest samples. Polynomial aggregation [4] holds similar idea with A-ARMA, while sensor nodes compute the polynomial coefficients as:

$$\mathcal{F}_{poly} \equiv \dot{x}_i = a_0 + a_1 i + \cdots + a_m i^m \tag{2}$$

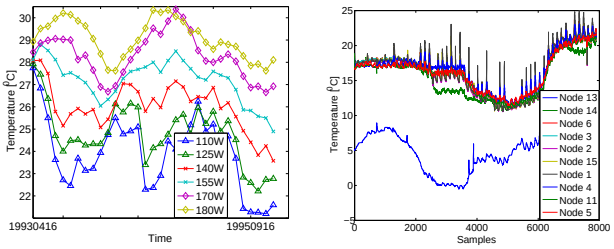
where $\dot{x}_i$ is recovered data, and $a_0, \cdots, a_m$ are coefficients of polynomial. Sensor nodes update m coefficients to sink replacing raw data to reduce the traffic. Sink recovers data from Eq. 2. All the above functions are belongs to forecasting aggregation functions [6].

III. Evolution analysis

In this section we describe our observations of similar evolution from real datasets first, and secondly show how sensor nodes model the evolution.

A. Observations from real datasets

We analyse several datasets to observe the evolution, here we present the two datasets we tested. First dataset is from Tropical Atmosphere Ocean (TAO) Project [7] (denoted as $\mathcal{TAO}$). The project focuses on monitoring the real-time ocean surface (The Pacific) temperature for understanding and prediction of $EINi\tilde{n}o$ and $LaNi\tilde{n}a$. Plotting the data from April 1993 to December 1995 per month, showed in Fig. 1(a), we observe that the nearby sensors generally express similar evolution, whereas the data are not the same (note that all the sensors are deployed in North Latitude 0^0). The temperature from the nearby location have the similar evolution with some fluctuation along time.



(a) Dataset $\mathcal{TAO}$. (b) Dataset $\mathcal{TMH}$.

Figure 1. Dataset $\mathcal{TAO}$ and $\mathcal{TMH}$, the observations of similar evolution behaviour.

Second dataset is from a custom project, which is temperature monitoring in a House (denoted as $\mathcal{TMH}$). The monitoring begins from January 2012 to July 2012, and nodes report every 10 minutes (about 26130 readings for every node). After 7903 samples, node 1 and node 6 suffer physical errors (no data reporting for a while), thus to data alignment, we use the first 7900 data to constitute $\mathcal{TMH}$, shown in Fig. 1(b).

B. Capture the evolution

From the above observations, we know that raw data often express similar evolution despite the data is not so close. Thus, how to capture the evolution is the present problem. Linear regression investigation [8] indicates that straight lines can be used to approximate time series, and the straight lines can be seen as evolution in our case. We know raw data of WSN can be seen as a set of (y, τ), in which y is the data reading and τ is corresponding data sequence. For example, temperature reading $(27.6, 10)$ denotes that the 10^{th} temperature value is 27.6^oC. Thus, linear regression in WSNs can be simply expressed as: $y = Rc \cdot \tau + \beta$, where Rc is the regression coefficient, and β is the incept. In our case, we use the Rc to describe the evolution.

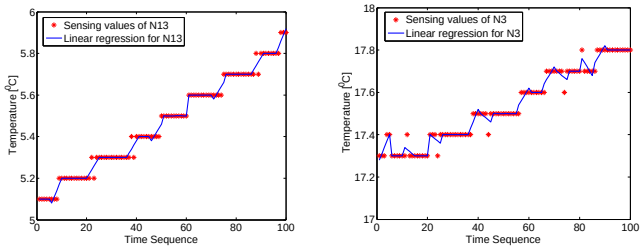
Considering the computation constrain of sensor node, we use piecewise linear regression, i.e., each 5 data as a segment (S_ι=5) to calculate Rc. Under one segment, regression coefficient Rc can be calculated as:

$$Rc_{S_{eg}} = \frac{\sum_{i=1}^{S_\iota}(\tau_i - \bar{\tau})(y_i - \bar{y})}{\sum_{i=1}^{S_\iota}(\tau_i - \bar{\tau})^2} \tag{3}$$

where $\bar{\tau} = \frac{1}{S_\iota}\sum_{i=1}^{S_\iota}\tau_i$, $\bar{y} = \frac{1}{S_\iota}\sum_{i=1}^{S_\iota} y_i$, S_{eg} denotes the current segment. With more segments, node generates a coefficient

vector $\vec{Rc} = [\cdots, Rc_{S_{eg}-1}, Rc_{S_{eg}}]$. In our experiment, we set $\|\vec{Rc}\| = 4$.

Fig. 2 verifies the piecewise liner regression result for given sensor nodes (N3 and N13 of dataset $\mathcal{TMH}$). We can see that the approximating line during one segment S_ι can fit the raw data well. Thus, in our context, evolution of sensor node is approximated by the regression coefficient.



(a) Raw data $1-100$ from N.13 and the linear regression result. (b) Raw data 1-100 from N.3 and the linear regression result.

Figure 2. 20-segment linear regression, using data of N3 and N13 from dataset $\mathcal{TMH}$, segment length S_ι is 5.

C. Measuring the similarity of evolutions

We use cosine similarity as a metric [9] to evaluate similarity. Cosine similarity is a measurement of similarity between two vectors, used to measure the angle between them. When two vectors are the same, the cosine similarity is 1; when the angle between two vectors is increasing, the cosine similarity is decreasing. Cosine similarity provide us a positive value ($(0, 1]$) to show the similarity. When the value is closer to 1, it means the nodes are in a more similar environment. The cosine similarity function between two vectors $\vec{Rc}^{\,i}$ and $\vec{Rc}^{\,j}$ is defined as:

$$Cs(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = \frac{\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,jT}}{\sqrt{(\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,iT})(\vec{Rc}^{\,j} \cdot \vec{Rc}^{\,jT})}} \tag{4}$$

where $\vec{Rc}^{\,T}$ states the transpose vector of $\vec{Rc}$, and Cs denotes cosine similarity. Note here, Cs function meets commutative law, i.e., $Cs(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = Cs(\vec{Rc}^{\,j}, \vec{Rc}^{\,i})$.

Therefore, to identify the similarity between the evolution, we provide following definition:

***Definition** 1:* We define the similarity function between two $\vec{Rc}$ as:

$$sim(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = \begin{cases} 1 & if \quad Cs(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) \geq th_{cs}, \\ 0 & otherwise. \end{cases} \tag{5}$$

where th_{cs} is a threshold determined by the application, which means if the cosine similarity of two $\vec{Rc}$ meets threshold, they are considered as similar. Since $\vec{Rc}$ represents the evolution of raw data, thus two nodes are similar if and only if their sim function is equal to 1. Note here, we define $\boldsymbol{th_{cs} > 0.5}$, which can avoid two orthogonal vectors grouping together.

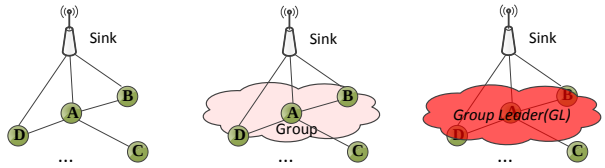
***Theorem** 1:* The similarity has transitive property. If $\vec{Rc}^{\,i}$ is similar with $\vec{Rc}^{\,j}$ and $\vec{Rc}^{\,k}$, thus $\vec{Rc}^{\,j}$ should be similar with $\vec{Rc}^{\,k}$, i.e.,

$$\begin{aligned} &\forall isim(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = 1 \;\&\; sim(\vec{Rc}^{\,i}, \vec{Rc}^{\,k}) = 1 \\ &\Rightarrow sim(\vec{Rc}^{\,j}, \vec{Rc}^{\,k}) = 1, \forall j, k \end{aligned} \tag{6}$$

This theorem is easily to be proved, due to the page limitation, we omit the proof process.

IV. Simba: Similar-evolution Based Aggregation

Fig. 3 demonstrates the two phases of *Simba*. Set-up phase (Fig. 3(b)) focuses on group forming, nodes group together due to similar evolution. Aggregation phase (Fig. 3(c)) focuses on executing data aggregation, group leader represents the group to update aggregated packet.



(a) Original network. (b) Set-up phase. (c) Aggregation phase.

Figure 3. The two phase in Simba, set-up phase is the processing of group forming, *GL* executes aggregation functions in aggregation phase.

A. Set-up phase

During set-up phase, each node calculates its $Rc_{S_{eg}}$ by Eq.3, and broadcasts it to neighbours (assuming the network is already connected). Node maintains a table to store the recent coming $Rc_{S_{eg}}$ and computes the similarity. When the similarity is higher than a threshold, marked as th_{cs}, the two nodes are considered having similar evolution.

As shown in Fig. 4, we take node A as an example to show the set-up phase. Supposing node A is adjacent with node B, C and D, after broadcasting $\vec{Rc}$, every node checks the table and calculates similarity. Table I shows an example of node A, where the column Nbr_ID is the neighbour ID, Rc is used to store $\vec{Rc}$, Cs and Sim are the values of Eq.4 and Eq.5 respectively.

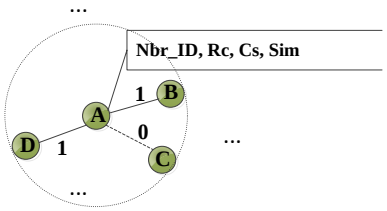


Figure 4. An example in set-up phase. The link labels correspond to the value of *sim* function.

Taking Tab.I as an example, node A finds node B and D are similar with itself. Considering **Theorem 1**, B and D

Nbr_ID	Rc	Cs	Sim
B	$\vec{Rc}^{B}$=[$\cdots$,$Rc^{B}_{S_{eg}-2}$, $Rc^{B}_{S_{eg}-1}$, $Rc^{B}_{S_{eg}}$]	Cs($\vec{Rc}^{A}$, $\vec{Rc}^{B}$)	1
C	$\vec{Rc}^{C}$=[$\cdots$,$Rc^{C}_{S_{eg}-2}$, $Rc^{C}_{S_{eg}-1}$, $Rc^{C}_{S_{eg}}$]	Cs($\vec{Rc}^{A}$, $\vec{Rc}^{C}$)	0
D	$\vec{Rc}^{D}$=[$\cdots$,$Rc^{D}_{S_{eg}-2}$, $Rc^{D}_{S_{eg}-1)}$, $Rc^{D}_{S_{eg}}$]	Cs($\vec{Rc}^{A}$, $\vec{Rc}^{D}$)	1

Table I. THE NEIGHBOUR TABLE OF NODE A.

are similar, even though they are not 1-hop neighbours. So node A multicasts to sink and node B,D that group {A,B,D} ({} notes the unit of one group) with timer $\frac{T}{x}$. Note that T is a timer, and x is the number of similar neighbours (i.e. $Sim = 1$ as defined by Eq. 5). Thus, a node having more similar neighbours will send notification more quickly than others. When a node receives the group notification from its neighbours, it marks its own transmission redundant, and then it cancels notification. This timer can avoid collision and redundancy, while note that irrespective of the strategy that a node follows, the result of the group remains unaffected. When a node has no similar neighbours, it does not have notification to sink. For sink side, if none of notification messages involves a node, it means this node forms a group with itself, it does not belong to any other groups.

B. Aggregation phase

In the aggregation phase of Simba, the group members share the aggregated information. It means the nodes having similar evolutions use the same coefficient to aggregate data. For example, node A and node B are in the same group, and we consider aggregation function is A-ARMA. Thus the model coefficient (φ and ϑ in Eq. 1) of A-ARMA computed by node A are also used to predict node B's data. In each group, only leader (*GL*) processes and transmits aggregated packet. With the group information in set-up phase, sink can recover *GL*'s data using aggregation function, and the data of other nodes can be retrieved easily.

1) Aggregating progress: At the beginning of aggregation phase, sink sends group result to one node of each group, and it select one node as the first group leader. In addtion, *GL* adds a field in its neighbour table to store the nodes in the same group. For example, node A receives the group from sink, it adds a list as A-B-D to assign the next leader, i.e., the next *GL* is node B, and node B will also store the list.

Briefly, aggregation progress is: node uses aggregation function to predicted data (or recovered data from sink view), when new raw data arrives, node compares the predicted data and raw data, if the error is beyond predefined threshold th_{err}, node transmits aggregated packet to sink; otherwise, node waits for new data to compare, and repeats the process. In Simba, every leader will process data as normal aggregation process. While when the current leader (e.g. node A) finds the accuracy exceeds th_{err}, it sends new aggregated packet to sink and the next leader (e.g. node B). The new *GL* (node B) will use this aggregated packet to predict data and compare with new raw data, and then repeats the process as last leader (node A). In this scheme, there is no effect from the size of group, all members of group are passively to wait for the instruction to execute aggregation function. From the list field in aggregated packet, group member becomes group leader in turn.

2) Recovering progress: For a group leader A, sink recover its data directly from the aggregation functions as (see Sec. II):

$$\widehat{y}^{A} = \mathcal{F}(y^{A}) \tag{7}$$

where $\mathcal{F}$ denotes the aggregation function, $\widehat{y}^{A}$ states the recover data of node A, and y^{A} is the raw data of node A. For A-ARMA and polynomial aggregation, sink recovers data from Eq. 1 and Eq. 2 respectively. For Average, sink uses $\widehat{y}$ as the average value for recover data. Suppose that node B is in the group of node A, and the difference between node A and node B is:

$$D_{AB} = \widehat{y}^{A} - \widehat{y}^{B} \tag{8}$$

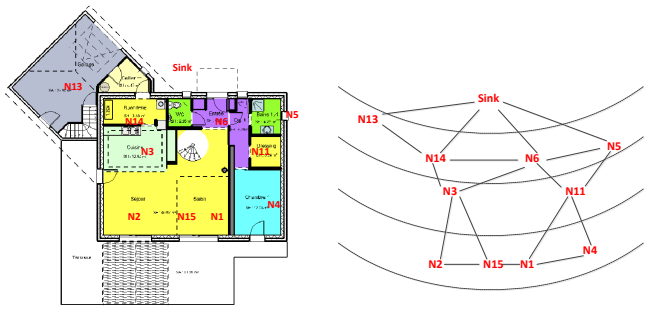
where D_{AB} marks the difference. Since the raw data maintain similar evolution, thus sink can simply recover data of node B. The data of node B can be recovered as:

$$\widehat{y}^{B} = \widehat{y}^{A} - D_{AB} \tag{9}$$

Therefore, from the Eq.7, Eq.8 and Eq.9, sink can recover all data for a group. Group leader is responsible for transmitting the aggregated packet to sink and next leader, and due to similar evolution, the group member can share the same aggregated packet. Similar evolution helps sink recover all nodes data and guarantees the recover accuracy. Moreover, the cooperation between nodes in the same group substantially reduces the group transmission as well as energy consumption.

V. PERFORMANCE EVALUATION

Based on the two datasets, we simulate the set-up phase using WSNet [5], an event-driven simulator for wireless networks, to show the group results. We simulate the aggregation phase using Matlab considering the $\mathcal{TMH}$ dataset. Note here, the network is connected before *Simba*, and we use GPSR [10] for data set $\mathcal{TMH}$, the node deployment and topology are shown in Fig. 5. For $\mathcal{TAO}$, all the nodes can be seen in a chain topology corresponding there location, and each node can connect with its 1-hop neighbours.



(a) The nodes deployment. (b) The schematic routing for $\mathcal{TMH}$.

Figure 5. The nodes topology of dataset $\mathcal{TMH}$.

A. Selection of similarity threshold th_{cs}

Similarity threshold th_{cs} is defined to check the similarity between different nodes, the higher th_{cs} is, more similar the evolution of values is. On one hand, a higher th_{cs} is desirable to maximize the similarity between nodes to keep accuracy. On the other, if it is too high, no nodes can be grouped together, there is no means for aggregation. Thus, choosing an appropriate th_{cs} is necessary for Simba.

We use A-ARMA(2,2) with Simba to investigate number of group and recover accuracy considering dataset $\mathcal{TMH}$. In Fig.6, we plot with y-axes on both two sides, where the left one corresponds to recovery error and the right one corresponds to group number. When similar threshold $th_{cs} = 1$, the group number is 10, it means every node forms a isolated group. Correspondingly, recover accuracy is highest (RMS$\leq$ 0.15) because every node only executes aggregation function by its own data. As decrease of th_{cs}, more nodes meet th_{cs}, they group together to execute A-ARMA(2,2), the accuracy is decreasing. When $th_{cs} = 0.75$, we have only 3 group in the network, while the accuracy is lowest. Thus, considering energy saving and accuracy, we chose 0.8 as the similarity threshold for dataset $\mathcal{TMH}$. Similarity threshold for dataset $\mathcal{TAO}$ can be calculated using the same method, which is 0.85. Normally, for dataset from in-door temperature readings, th_{cs} can be setting around 0.8, which can keep the accuracy of recovery data and group size.

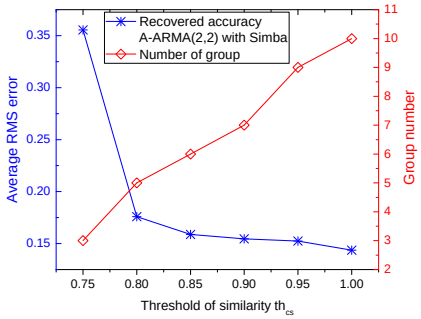


Figure 6. The influence of threshold of similarity th_{cs} on recover accuracy and group result, based on dataset $\mathcal{TMH}$ (error threshold $th_{err} = 0.03$).

B. Accuracy and Energy issues

A good aggregation scheme should keep high recover accuracy and save more energy. In this section, we investigate the accuracy and energy issues of Simba. First, we analyse the recover accuracy of A-ARMA, polynomial aggregation and average with and without *Simba* within dataset TMH.

In Fig.7, we show the detail recover accuracy of each node of 3 original aggregation functions, and comparing with situation of Simba. If node has no group, every node works on the original functions, there is no change for recover accuracy, such as $N13$, $N14$ and $N5$. For the nodes grouped together, we can see that Simba indeed introduces some error but not so obviously. This is because only *GL* operates aggregation function on its own data, the others data are recovered based on similar evolution. Besides, there is an error threshold th_{err}

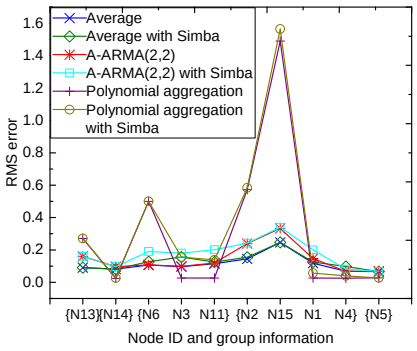


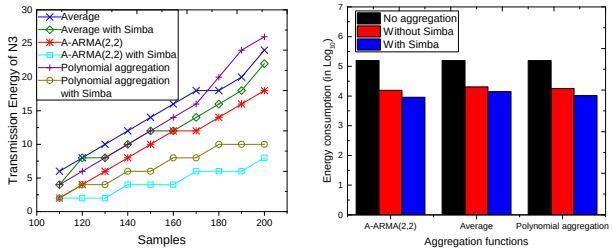
Figure 7. The recover accuracy with and without Simba (nodes ID in {} in x-axis are group members). Here, error threshold $th_{err} = 0.03$.

makes nodes to respect accuracy. Thus, the margin between original functions and Simba is not so big. For example, the average RMS error of 10 nodes for A-ARMA(2,2) is 0.1437, and error for A-ARMA(2,2) with Simba is 0.1758. That is to say, with Simba, nodes respect the accuracy in a group, which can guarantee the final fidelity.

In terms of energy, we detail the energy consumption for one node firstly, and then demonstrate the energy consumption of the whole network. Takes N3 as an example, which is one member of the group {N3,N6,N11}. Assuming energy consumption of transmitting an aggregated packet is 1, and the energy for reception is 1. As samples arrives, we compare transmission energy of aggregation functions with and without Simba. Calculating transmission energy begins from aggregation phase (removing the 100 samples for Set-up duration). Fig. 8(a) plots the energy consumption of N3 from data sequence 100 to 200, we can see that whatever the aggregation function is, with Simba, the energy consumption decreases. This is because Simba moves the energy consumption from a node to a group. N3 belongs a group {N3,N6,N11}, and the group is represented by leader. When N3 is current *GL*, it needs to transmit aggregated packet; when N3 is not the current *GL*, it keeps silence to save energy. Thus with Simba, the group member can save more energy than they originally do. More specifically, Fig.8(b) demonstrates the total energy consumption for the whole network (each node generates 7800 packets) in situation of no aggregation, without Simba and with *Simba*. We can see that the whole energy consumption is reduced by the situation with Simba (scale in Log). Whatever the functions are, Simba helps them to save energy, and at least saves 30% more than the original functions do. For A-ARMA(2,2), Simba achieves a 41% reduction on energy consumption. Comparing with the situation of no aggregation, Simba can save energy around 94%, 91% and 95% for 3 aggregation function respectively.

C. Comparative analysis

To the best of our knowledge, few aggregation works focus on data evolution, similar works all pay attention on raw data. We select the work using raw data to cluster nodes as the state-of-the-art benchmark for Simba, characteristic correlation [2]. To compare with Simba, we simulate [2] in Matlab with dataset $\mathcal{TMH}$, and nodes are deployed as the hierarchy in Fig. 5(b).



(a) Energy consumption for N3 (calculating every 10 data).

(b) Total energy comparison in different aggregation functions (Log scale).

Figure 8. Energy comparison from one node to the whole network.

We use 0.5^0C and 1^0C as "categorizing range" respectively. Actually, this "categorizing range" is error threshold th_{err} in Simba, thus we also set $th_{err} = 0.5^0C$ to compare with characteristic correlation. The first 100 raw data of dataset are used to make cluster stable (which is also proposed in [2], with enough raw data, the cluster keeps stable and then node is able to do aggregation). Average is used as aggregation function in Simba and characteristic correlation. We calculate RMS error (between recovery value and real value) for every 10 data, to show the error from two aggregation methods.

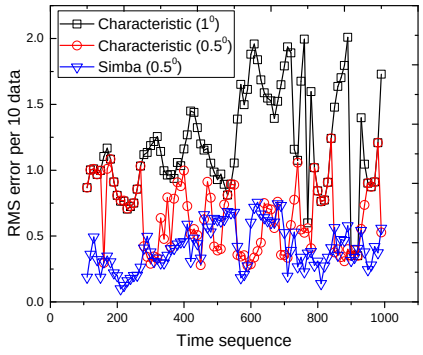


Figure 9. Comparison analysis between characteristic correlation and Simba, considering error threshold as 0.5^0C and 1^0C.

Fig. 9 demonstrates the RMS error per 10 data for characteristic correlation and Simba, we can see Simba performs lower RMS error than characteristic correlation. Moreover, with different "categorizing range" (0.5^0C and 1^0C), characteristic correlation shows very dynamic RMS error. This is because characteristic correlation relies on raw data, this method groups the nodes holding similar raw data, but raw data often show dynamic properties. For Simba, it uses evolution to group nodes together, the evolution is more stable than raw data. Besides, in characteristic correlation, sink retrieves the average values as all cluster members data, it does not consider the difference between nodes. While for Simba, sink uses the mean difference to recover data, this can guarantee the accuracy further.

VI. Conclusion

We propose *Sim*ilar-evolution *B*ased *A*ggregation, a raw data-independent aggregation, to take benefits taken from similar evolution in this paper. Our simba works as: nodes use linear regression to catch the evolution and group with the similar neighbours together. The nodes in one group alternatively send the aggregated packets to sink. Each node uses a few packets to transmit, and sink can recover the whole data for a group due to similar evolution. We can see, similar evolution give nodes the opportunity of cooperating with each other when executing aggregation function. The experiments results show that Simba use less aggregated packets to get high fidelity recover value, and Simba can save at least 30% more energy than original aggregation functions do. Comparing with situation of no aggregation, Simba can save energy more than 91%. In the future, we will extent our protocol within more dynamic situation, e.g., dynamic evolution and topology change.

Acknowledgement

We would like to thank Tropical Atmosphere Ocean (TAO) project for its dataset. This work was particularly supported in part by the INSA Lyon BQR ARBRE project.

References

[1] I. Solis and K. Obraczka, "Isolines: efficient spatio-temporal data aggregation in sensor networks," *Wireless Communications and Mobile Computing*, vol. 9, no. 3, 2009.

[2] H. Li, V. Pandit, A. Knox, and D. Agrawal, "A novel characteristic correlation approach for aggregating data in wireless sensor networks," in *2013 IEEE WoWMoM*, Madrid,Spain, June 2013.

[3] J. Lu, F. Valois, M. Dohler, and M.-Y. Wu, "Optimized data aggregation in wsns using adaptive arma," in *SENSORCOMM*, Venice,Italy, July 2010.

[4] S. Ozdemir, M. Peng, and Y. Xiao, "Prda: polynomial regression-based privacy-preserving data aggregation for wireless sensor networks," *Wireless communications and mobile computing*, vol. 15, no. 4, 2015.

[5] INRIA, "WSNet simulator." [Online]. Available: http://wsnet.gforge.inria.fr

[6] J. Cui and F. Valois, "Data aggregation in wireless sensor networks: Compressing or forecasting?" in *IEEE WCNC*, Istanbul, Turkey, Apr. 2014.

[7] T. project, "Temperature data." [Online]. Available: http://www.pmel.noaa.gov/tao/data_deliv/deliv.html

[8] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2012, vol. 821.

[9] S. Van Dongen and A. J. Enright, "Metric distances derived from cosine similarity and pearson and spearman correlations," *arXiv preprint arXiv:1208.3145*, 2012.

[10] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Gradient broadcast: A robust data delivery protocol for large scale sensor networks," *Wireless Networks*, vol. 11, no. 3, 2005.