Improving Learning Automata-based Routing in Wireless Sensor Networks

E. Ahvar
Information Technology
and Communication Department
Payame Noor University, Iran

M. Yannuzzi, R. Serral-Gracià
E. Marín-Tordera, X. Masip-Bruin
Advanced Network Architectures Lab
Vilanova i la Geltrú, Spain

S. Ahvar
Electrical and Computer
Engineering Department
Isfahan University of Technology, Iran

***Abstract*—Recent research in the field of Wireless Sensor Networks (WSNs) has demonstrated the advantages of using learning automata theory to steer the routing decisions made by the sensors in the network. These advantages include aspects such as energy saving, energy balancing, increased lifetime, the selection of relatively short paths, as well as combinations of these and other goals. In this paper, we propose a very simple yet effective technique, which can be easily combined with a learning automaton to dramatically improve the performance of the routing process obtained with the latter. As a proof-of-concept, we focus on a typical learning automata-based routing process, which aims at finding a good trade off between the energy consumed and the number of hops along the paths chosen. In order to assess the performance of this routing process, we apply it on a WSN scenario where a station S gathers data from the sensors. In this typical WSN setting, we show that our combined technique can significantly improve the decisions made with the automata; and more importantly, even though the proof-of-concept particularizes somehow the automata and their behavior, the technique described in this paper is general in scope, and therefore can be applied under different routing methods and settings using learning automata.**

I. Introduction

In the past few years, a plethora of research works have shown the advantages of using learning automata in the field of Wireless Sensor Networks (WSNs). The literature in the topic is large and diverse, but a good sample of the strengths of applying learning automata in the context of WSNs can be found in [1], [2], [3], [4], and [5].

While the theory of learning automata is well understood, in practice, a designer that aims at exploiting the functionality of a learning automaton within a sensor faces multiple challenges, some of which are not easily solvable. On the one hand, most of the relevant problems in the context of WSNs have a multi-objective nature, which poses complex challenges during the design phase of the automata. For instance, the designer needs to define a probability set, which needs to capture somehow the chances to succeed in the selection of one candidate that best fits the multi-objective problem to be solved. On the other hand, the probability set needs to be maintained and updated upon receiving feedback from the environment, and this is typically handled on the basis of a reward/penalty scheme which also needs to be defined and tuned. In general, these decisions strongly condition the behavior of the automata, and therefore, the performance ultimately obtained with them.

This work was supported in part by the Spanish Ministry of Science and Innovation under contract TEC2009-07041, and by the Catalan Government under contract 2009 SGR1508.

In this paper, we focus on a typical WSN setting, where the routing process running in each sensor is supported by a learning automaton. In this framework, we target a twofold objective, which is to find a good trade off between the energy consumed by the sensors and the number of hops along the paths chosen by these latter.

Overall, this paper makes the following contributions. First, we question somehow the effectiveness of the automata to succeed in the achievement of the set objectives. To this end, we put ourselves in the shoes of the designer and describe a typical learning automata-based routing process. After that, we propose a very simple technique which can work in combination with the learning automata, and show that the combined routing technique can actually outperform the automata. We show that with our synergetic and simple routing scheme, the energy savings are significant. We also show that the combined technique works much better than each of its parts in isolation.

The rest of the paper is organized as follows. In Section II, we introduce the WSN model used, including the processes for discovering neighbors and for maintaining the data that will be used by the automata. Section III describes a typical learning automata-based routing scheme for a WSN. Section IV describes the combined technique proposed in this paper, which is later on evaluated in Section V. Finally, Section VI concludes the paper.

II. The Network Model

Let us consider a WSN composed of a set of sensor nodes and a station S, where the latter gathers data from the sensors. The station S may be either a fixed or a mobile node, and it is assumed to be the one connecting the WSN to a communications infrastructure through which a user can access the collected data. All the packets forwarded in this sensor network are tagged with a *message type*. This is to distinguish the broadcast messages that need to be flooded along the network from unicast messages encoding the events sent from the sensors toward the station S.

In this paper, we shall focus on a setting in which the routing process running in each sensor is supported by a learning automaton. In these cases, the automata in the sensors will typically compute and maintain a set of probabilities, which will determine the selection of the best possible neighbor during the packet forwarding process. As mentioned above, the automata present in our WSN will consider two metrics while computing these probabilities: energy levels and the number of hops to reach the station S—the details about these metrics and their role in the computation of the probabilities will be described later on in Section III.

In order to supply the required information for the operation of the learning automata, every node i maintains a basic "Neighbor List", which consists of four fields storing the following data associated to each neighbor: 1) the ID of the neighbor, j_{ID}, with $j = 1, \dots, N_i$, where N_i denotes the number of neighbors of node i, and thus the size of the list; 2) the energy level $\mathcal{E}_j(t)$ at time t of neighbor j (except for $j = S$, since the station's energy is assumed to be unlimited compared to that of the sensors); 3) the hop count $H_j(t)$ through neighbor j to the station S at time t; and 4) the probability $P_j(t)$ associated with neighbor j as computed by the learning automaton in node i. Note that the routing protocols supported by learning automata usually make forwarding decisions based on the highest probability associated to its neighbors, and thus this is the approach followed in this paper.

In this scenario, the processes for neighbor discovery and population of the Neighbor Lists can be summarized as follows. When a node i receives a message for the first time from a neighbor j, this message produces a new entry in its Neighbor List—note that the neighbor j could perfectly be the station S if node i is within the radio reach of S. Each message piggybacks the data that is needed to populate and to update the fields in the Neighbor List. Indeed, the messages distributed through the WSN are processed at each hop j, so the message originally sent by the source is piggybacked with the energy level of node j, and the list of hop IDs from j to S (including j_{ID}). Observe that the latter has a twofold purpose. First, to serve for the hop count; and second, similarly to the case of the Border Gateway Protocol (BGP), it provides a straightforward mechanism to prevent forwarding loops in the sensor network. When node i receives the message, it swaps node j's energy with its own energy level and also appends its own ID to the list of hops before forwarding the message to its neighbors. In stationary state, every node in the network will know the energy levels of their neighbors and the hop counts reported from these latter to the station S.

III. Learning Automata and Update Strategy

Consider a routing process steered by a learning automaton running on each sensor i. We assume that the decision criterion of the automaton in node i considers both the energy levels and the hop counts reported by its neighbors. Observe that these are two traditional goals for decision making when exploiting learning automata for routing purposes in WSNs. Indeed, the objective of a typical automaton could be to balance the load among the different sensors with a twofold goal: i) avoid that the sensors run out of battery; ii) while keeping the routes toward the destinations relatively short.

To this end, we consider a learning automaton that operates as follows. Based on the energy levels and hop counts stored in the Neighbor List, the automaton in i associates a probability $P_j(t)$ to each neighbor j, which is also stored in the Neighbor List and which we assume is computed according to expression (1).

$$P_j(t) = \frac{1}{2}\left(\frac{\mathcal{E}_j(t)}{\sum_{m=1}^{N_i}\mathcal{E}_m(t)} + \frac{1/H_j(t)}{\sum_{m=1}^{N_i} 1/H_m(t)}\right) \; j \leq N_i \quad (1)$$

where $\mathcal{E}_m(t)$ is the energy level advertised by neighbor m, N_i is the size of node i's Neighbor List, $H_m(t)$ is the hop count advertised by neighbor m to the station S (including node m), and the sums in the denominators represent the terms to normalize the probabilities. The rationale of using (1) is that it provides a reasonable balance between energy and the hop count, although it comes at the cost of the potential recomputation of the probabilities right after receiving a message from a neighbor, since $\sum_{j=1}^{N_i} P_j(t) = 1, \; \forall \; t$. Note, however, that this is always the case when the energy states are involved in the computation of the probabilities.

It is worth highlighting that since the energy levels are reported (piggybacked) by the neighbors, in our model the values $\mathcal{E}_m(t)$ that are stored in the Neighbor List are already adjusted considering the energy that is actually required to transmit the packet from node m to node i.

Under this configuration, we assume that the learning automaton in node i will decide to forward the packets to the neighbor with the highest probability.

A. The Update Strategy

In a nutshell, any learning automaton will make a decision based on the information at hand (e.g., select the next-hop from a list of candidates and send data packets to it), and will then wait for feedback about the outcome of its selection or action. The materialization of this feedback process basically consists in the update of the probability set, often in the form of a penalization or rewarding strategy, which is what will ultimately define the behavior of the automaton.

In our case, we consider the feedback process shown in Fig. 1. The automaton in node i selects the node with the highest probability as the next-hop (node j in this case), and then it sends the packets to it. When node j receives the packets, it updates the energy level and hop count of node i in its Neighbor List, and it may also update the probability $P_i(t)$ depending on the energy and hop count reported by node i to the station S. As shown on the left-hand side of Fig. 1, all the other neighbors in the range of i can overhear the response and potentially perform the same updates as j, though discarding the packets right after processing them. When j forwards the packets to k (see the right-hand side of Fig. 1), node i and the rest of the neighbors of node j can overhear the transmission and thereby update the energy level and hop count of the

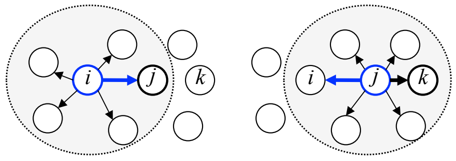


Fig. 1. Updating the energy levels, the hop counts, and the probabilities through piggybacking and overhearing techniques. (Left-hand side) The neighbors of node i perform the updates. (Right-hand side) Idem for node j.

latter. Overall, we consider a basic setting in which the learning automata can efficiently update the probabilities in their Neighbor Lists based on simple piggybacking and overhearing techniques.

In the example shown in Fig. 1, if the metrics in the feedback received from node j are acceptable, then node j is rewarded by the learning automaton in i, and the probability associated to j is increased in node's i Neighbor List. Otherwise, j is penalized and its probability is decreased. To formalize this, let $x = \{x_j\}$ be the behavioral vector, where we define:

$$x_j = \frac{\mathcal{E}_j(t)}{<\mathcal{E}_i(t)>} \frac{<H_i(t)>}{H_j(t)} \tag{2}$$

in which $<\mathcal{E}_i(t)> = \sum_{m=1}^{N_i} \mathcal{E}_m(t)/N_i$ represents the average energy of all neighbors of node i, and $\mathcal{E}_j(t)$ stands for the energy level obtained from j. Likewise, $<H_i(t)>$ represents the average hop count of all neighbors of i to the station S, while $H_j(t)$ denotes the hop count to S reported by node j. Depending whether the behavioral variable x_j is below or above these averages, the learning automaton in i will reward or penalize node j using the functions $\alpha(x_j)$ and $\beta(x_j)$, respectively (these functions will be described in more detail in the next few paragraphs). This is a standard approach in learning automata theory, and in our model we consider four behavioral zones for rewarding or penalizing a neighbor, which are captured through the incentive function $\mathcal{I}(x_j)$:

$$\mathcal{I}(x_j) = \begin{cases} \beta(x_j) & \text{if} \quad x_j < 1 & \text{(max. penalty)} \\ \beta(x_j)/2 & \text{if} \quad x_j = 1 & \text{(half penalty)} \\ \alpha(x_j)/2 & \text{if} \quad 1 < x_j < 1.5 & \text{(half reward)} \\ \alpha(x_j) & \text{if} \quad x_j \geq 1.5 & \text{(max. reward)} \end{cases} \tag{3}$$

We proceed now to describe in more detail the typical incentive mechanism outlined above.

Reward function—The reward function $\alpha(x_j)$ is part of the update strategy, and it is used to increase the priority of the neighbors with more possibilities to deliver the packets. In our model, the function is computed using $\alpha(x_j) = \lambda_\alpha + \delta_\alpha x_j$, where λ_α is the minimum reward granted to a well-positioned node, and δ_α is the limiting factor for the reward.

Penalty function—Similarly, we use $\beta(x_j) = \lambda_\beta + \delta_\beta x_j^{-1}$, where analogously to the reward mechanism, λ_β is the minimum penalty, and δ_β is the limiting factor. Note that in (3), the better (worse) the energy–hop count relationship x_j the greater the reward (penalization) assigned to node j.

Once the energy and hop count metrics are updated for node j, the learning automaton in node i will proceed to update the probabilities of its N_i neighbors based on equations (4) and (5). This is a standard procedure, where (4) applies for the rewarding cases, i.e., either when $\mathcal{I}(x) = \alpha(x)/2$ or $\mathcal{I}(x) = \alpha(x)$, while (5) corresponds to the penalization cases, that is, when $\mathcal{I}(x) = \beta(x)$ or when $\mathcal{I}(x) = \beta(x)/2$.

$$\begin{cases} P_j(t_{n+1}) = P_j(t_n) + \mathcal{I}(x_j)(1 - P_j(t_n)) \\ \\ P_m(t_{n+1}) = (1 - \mathcal{I}(x_j))P_m(t_n) \quad \forall\, m \neq j \end{cases} \tag{4}$$

$$\begin{cases} P_j(t_{n+1}) = (1 - \mathcal{I}(x_j))P_j(t_n) \\ \\ P_m(t_{n+1}) = \dfrac{\mathcal{I}(x_j)}{N_i - 1} + (1 - \mathcal{I}(x_j))P_m(t_n) \quad \forall\, m \neq j \end{cases} \tag{5}$$

After describing this rather standard application of learning automata in the context of routing in WSNs, we proceed to describe first the combined routing technique, and then to assess the performance gain obtained with this synergetic strategy.

IV. Combined Routing Technique

In this section, we describe a very simple yet effective technique that, as we shall show later in Section V, can significantly improve the performance obtained with a learning automaton. The basics of this technique are illustrated in Fig. 2, and it works as follows. When a sensor i receives a packet, the routing process is invoked, and the Learning Automaton (LA) selects the ID of the next-hop, $ID^{(LA)}$, based on the probability set stored in the Neighbor List. In parallel, a separate module with access to the Neighbor List also selects the next-hop based on an alternative criterion. Due to its effectiveness and simplicity, one possibility is to rely on one of the criteria that is widely used in operational networks, namely, a compound metric. In this paper, we follow this approach, which is represented as the Compound Metric (CM) module in Fig. 2. Thus, this module selects in parallel the next-hop, which we denote as $ID^{(CM)}$. Figure 2 shows that a Decision Maker module selects the next-hop that will be finally used.

The main contribution of this paper is that we show that this combined selection outperforms each of the two possible selections working in isolation. Indeed, we show that even when the majority of the decisions made by the Decision Maker module fall on LA side, the small fraction of decisions that fall on the CM side have a significant impact on the energy and the lifetime of the sensors.

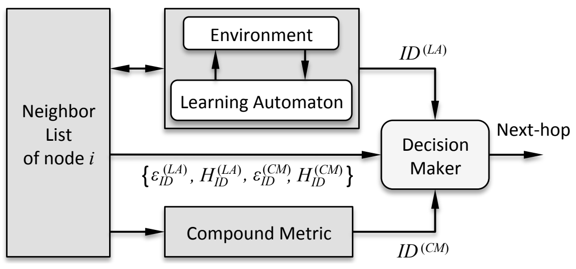

Neighbor List of node i
Environment
Learning Automaton
$ID^{(LA)}$
Decision Maker
Next-hop
$\{\varepsilon_{ID}^{(LA)}, H_{ID}^{(LA)}, \varepsilon_{ID}^{(CM)}, H_{ID}^{(CM)}\}$
Compound Metric
$ID^{(CM)}$

Fig. 2. Combined Decision Technique.

This combined technique has also two features that are worth highlighting. On the one hand, in Figs. 3 and 4 we shall show that the additional energy consumption posed by the parallelized processing more than pays for itself. On the other hand, irrespectively of the behavioral definition of the learning automata or the parallelized module used, our results suggest that even very simple synergetic techniques, such as the one described here, are sufficient to improve the routing decisions made by a learning automaton.

In this paper, the compound metric chosen is the following:

$$C_{ij}(t) = \begin{cases} \omega\mathcal{E}_j(t)\left(1 - \dfrac{H_j(t)}{H^{max}}\right) & \dfrac{\mathcal{E}_j(t)}{<\mathcal{E}_i(t)>} > 1 \\ 0 & \dfrac{\mathcal{E}_j(t)}{<\mathcal{E}_i(t)>} \leq 1 \end{cases} \quad (6)$$

where ω is a control parameter that simply limits the energy factor $\omega\mathcal{E}_j(t)$ to the interval [0,1] $\forall$ j, and H^{max} is the maximum number of hops reported among all the neighbors of node i. Note that the energy threshold used in (6) is to remove from the selection process the neighbors j whose energy level is below the average. The rationale of using (6) is that again, it produces a good balance between energy and the hop count in the form of a simple compound metric.

The processing of the Decision Maker module is summarized in Algorithm 1. Note that if the selections made by the LA and CM modules match, then the node selected is chosen as the next-hop. Otherwise, the Decision Maker runs a basic sequence of tie-breaking rules until the next-hop is selected.

Algorithm 1: The Decision Maker

```
Inputs : $\{ID^{(LA)}, \mathcal{E}_{ID}^{(LA)}, H_{ID}^{(LA)}\}, \{ID^{(CM)}, \mathcal{E}_{ID}^{(CM)}, H_{ID}^{(CM)}\}$
Output: next-hop node
1 foreach packet to be forwarded do
2     if $ID^{(LA)} == ID^{(CM)}$ then
3         send the packet to the selected neighbor;
4     // If IDs do not match then run tie-breaking rules;
5     else if ($\mathcal{E}_{ID}^{(LA)} > \mathcal{E}_{ID}^{(CM)}$) && ($H_{ID}^{(LA)} < H_{ID}^{(CM)}$) then
6         choose $ID^{(LA)}$ as the next-hop;
7     else if ($\mathcal{E}_{ID}^{(LA)} < \mathcal{E}_{ID}^{(CM)}$) && ($H_{ID}^{(LA)} > H_{ID}^{(CM)}$) then
8         choose $ID^{(CM)}$ as the next-hop;
9     else choose the one with the highest energy;
```

V. Performance Evaluation

In this section, we test the performance obtained with the combined routing technique depicted in Fig. 2. To this end, we analyze the performance of the Learning Automata (LA) and the Compound Metric (CM) modules separately, and compare the results to the case when they operate together. Hereafter, we shall refer to the joint operation to as the Combined Technique (CT). In order to widen the analysis, we also compare the performance of the LA, CM, and CT routing processes with state of the art alternatives that especially target the optimization of energy and the hop count in flat sensor networks. In the evaluations shown in this paper, we considered the Maximum Available Power (MAP) [6], and the Minimum Hop Routing (MHR) [7] schemes. These fall under the same category of the routing scheme described in this paper, i.e., under the family of source-initiated routing protocols. The assessments described here were carried out by means of simulation, and the tool chosen for running the tests was the Glomosim simulator developed by UCLA [8].

A. Simulation Model

The performance of the different routing schemes is evaluated using a surface of $1000m^2$, with a radio range of $198.5m$ and a TX-power of 5.0 dBm. The available bandwidth used was $2Mbps$, in a network based on IP over 802.11. To ensure the repeatability of the tests under the same conditions, in our first round of simulations all the nodes involved were randomly placed on the surface, and we used the same positioning of the sensors while evaluating the remaining routing schemes. Under this configuration, we performed a total of five different tests separated into two different scenarios which are described below. Each test was repeated 10 times, and the results shown in Figs. 3 and 4 correspond to the averages obtained.

In the different experiments carried out, we also evaluated the effects of the amount of nodes in the network. Thus, we ran all the simulations in three sets of rounds with 1000, 1200, and 1400 nodes. For each round, the procedure for generating traffic in the network works as follows. After the discovery phase (i.e., once every node knows the IDs, the energy levels, and the hop counts of its neighbors), a sensor is randomly selected every 10 seconds, which initiates a packet transmission to one of its neighbors toward the station S.

Scenario I—In this scenario, we assume a critical situation, where the energy level for transmission mode is very low (0.00055 mW in our case). Under these conditions, we evaluate the different routing schemes considering three different tests:

- *Test 1: Time until the first node runs out of battery—* This test is one of the indicators of the effectiveness of the routing schemes in terms of energy management. In general, those with the capacity to balance the energy consumed should last longer without node failures.
- *Test 2: Number of nodes that run out of battery—*This test computes the total number of sensors that fail for each routing scheme during a simulation period of 2 hours. This simulation provides an indicator of the capacity of the routing schemes for saving energy.
- *Test 3: Fraction of active neighbors of the station S at the end of the simulation—*This test shows the ability of the different routing schemes to keep the station S connected. As in the case of *Test 2*, the simulation period for this test is of 2 hours.

Scenario II—In this scenario, the energy levels of the sensors are set sufficiently high so as to avoid experiencing node failures during the simulation runtime. Our goal in this case, is to compare the fairness in terms of energy consumption. In order to avoid bias in the comparison, we ensure that all the routing schemes assessed transmit the same amount of data, and that this occurs without node failures. We carry out two tests to examine how the routing schemes save and manage energy in regular operation mode.

- *Test 4: Average energy consumption*—This test provides another indicator of which routing scheme is more efficient at managing energy.
- *Test 5: Variance in the remaining energy levels for the neighbors of the station S*—This test will allow to examine which routing scheme is able to perform better energy balancing among the nodes close to the station.

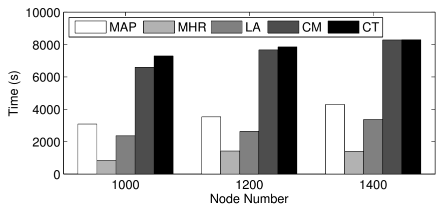
In summary, for the two considered scenarios, we have strongly focused on the assessment of the energy aspects of the different routing schemes, since this is the most important metric in many practical WSN settings. In particular, for source-initiated routing protocols, where the sensors basically need to report and distribute new events to the station S, other metrics such as delay, hop count, etc., are often less critical than the batteries lifetime. In these cases, the hop count is usually used to avoid that the energy balancing strategy ends up yielding excessively large routes to reach the station S.

B. Simulation Results

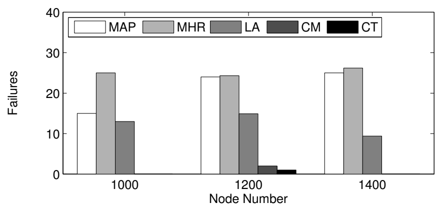
1) Scenario I: Fig. 3(a) details the outcome of *Test 1*, i.e., the time elapsed until the first node depletes its battery. These results provide a first indicator of the performance obtained with the five different routing schemes under test. As it can be noted, when using the LA in isolation, the first sensor fails after ~ 40 to ~ 60 minutes depending on the number of nodes present in the network. It can also be observed that the instant in which the first node runs out of battery is relatively similar for LA and MAP, and significantly shorter for MHR. However, this performance can be dramatically improved through the utilization of a simple compound metric (CM), with which the first node failure occurs after almost 2 hours. Although at first sight the results shown in Figs. 3 and 4 could demystify in part the effectiveness of using learning automata in the route selection process, it is worth discussing a number of issues in order to avoid reaching premature conclusions.

Firstly, even though in Section III we have made an effort to describe a typical learning automata-based routing scheme, the computation of the probabilities in (1) as well as the incentive function in (3) are anyway particular, and thus condition the performance ultimately obtained with the LA. Secondly, the compound metric chosen in (6) is also particular; and thirdly, the WSN model and its application are also somehow particular, all of which bias the comparisons made in Figs. 3 and 4. Despite this, it is worth noting that any designer exploiting learning automata in the routing process of a sensor network will face the same kind of decisions, including deciding on how to compute the probabilities, their update, and so on.

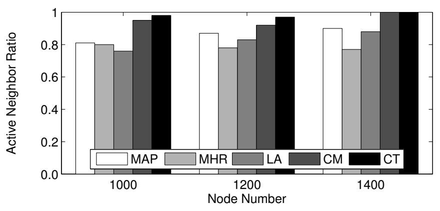
Therefore, the relevance of our results resides in the qualitative analysis rather than in the quantitative one. Indeed, instead of comparing the particular values obtained in each figure, the most important conclusions that can be extracted from Figs. 3 and 4 as a whole are basically the following. The results show that even the addition of a very simple module can help the designer of a routing process supported by a learning automaton to significantly improve its performance, and more importantly, the combined operation might work better than each module in isolation. These are in essence the main con-



(a) *Test1:* Time until the first node runs out of battery.

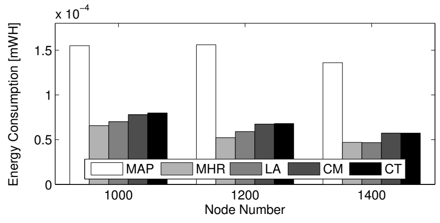


(b) *Test2:* Number of nodes that run out of battery.

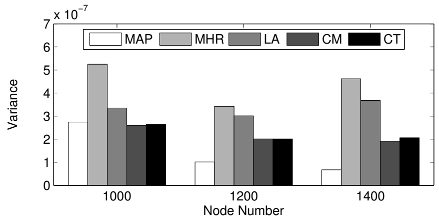


(c) *Test3:* Fraction of active neighbors of S.

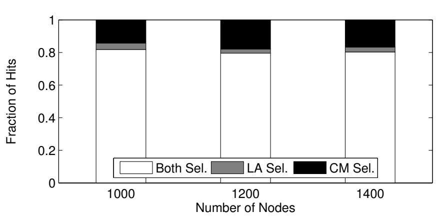
Fig. 3. Tests results for Scenario I.



(a) *Test4:* Average energy consumption.



(b) *Test5:* Variance.



(c) Decision Ratio.

Fig. 4. Tests results for Scenario II.

clusions that can be drawn from our results, and hence our main contributions in this work.

These assertions are supported by Figs 3(b) and Fig 3(c), where we detail the total amount of nodes that ran out of battery, and the fraction of active neighbors of the station at the end of the simulations, respectively. From Fig. 3(b), it can be noted that the number of nodes that fail with the CT scheme is minimum compared to the rest. Indeed, for 1000 and 1400 nodes, the CM and CT schemes show no failures at all. The results observed in Fig 3(c) are consistent with these findings, since the CM and CT schemes outperform the other alternatives. Also note that, for all the tests shown in Scenario I, the CT scheme performs always better than the LA and CM separately.

2) Scenario II: In this second scenario, the focus resides on the evaluation of the fairness in terms of energy consumption. As mentioned before, in this case the batteries are not in a critical state, and thus, we can examine the evolution of the energy consumed by the different routing schemes (in mWH) as well as their variance over time.

To this end, in Fig. 4(a) we detail the average energy consumption in mWH for the different routing schemes. As it can be observed, the scheme with the highest energy consumption on average is MAP, while the most efficient on average is MHR, with all the alternatives proposed in this paper in-between of these two. However, the variances shown in Fig. 4(b) reflect the fact that MAP shows a very low variance in general, while MHR shows the largest one. This is because, MAP provides a fairer distribution of energy consumption (it is a balancer scheme), whereas MHR aims at routing over the least number of hops and thus tends to consume more energy over a subset of given nodes. By examining the variances, we observe that the CT scheme again provides a fairer distribution of the energy consumed than the LA, though this better balancing comes at the cost of a slightly higher consumption of energy on average.

To conclude, Fig. 4(c) shows the breakdown of decisions made by the Decision Maker, indicating the fractions when the selection coincides with the one made by the LA, by the CM, as well as the fraction in which both agree in the selection of the neighbor. More precisely, the legends in Fig. 4(c) indicate the following: i) Both Sel.: means that the LA and CM modules in Fig. 2 yield the same neighbor ID (steps 2 and 3 in Algorithm 1); ii) LA Sel.: means that the Decision Maker prefers the neighbor chosen by the automaton (steps 5 and 6 in Algorithm 1); while iii) CM Sel.: means that the Decision Maker prefers the neighbor chosen by the CM module (steps 7 and 8 in Algorithm 1). Note that for 1400 nodes, the LA module provides around 85% of the neighbor selections (i.e., Both Sel. plus LA Sel.), however, Fig. 4(b) shows that the variance obtained with LA is large, and thus the resulting distribution of energy consumption is quite unfair. Conversely, with CT the variance is only about 50% of that of the LA scheme, which suggests that even a small fraction of decisions (around 15% in this case) can have a significant impact on the degree of fairness in energy consumption.

VI. Conclusion

This paper addresses the performance aspects of an increasingly used solution in the field of Wireless Sensor Networks (WSNs), namely, the learning automata. Even though many recent proposals leverage on the well-known strengths of learning automata, in this work we question somehow the effectiveness and performance obtained with them in the context of WSN routing. We have shown that the designer of an automaton faces multiple decisions and complex challenges, which can have a profound impact on the performance ultimately obtained with the automaton. In this scenario, we have described a very simple yet effective solution that can work in combination with the automata, and shown that our combined routing technique can significantly improve the performance achieved with them. Our analysis suggests that the decision on how to compute the probability set in the automata is far from trivial, especially, when the problem addressed has a multi-objective nature. We have also found that setting the policy for updating the probability set is really challenging, since minor tweaking might derive in over-penalizing of over-rewarding some neighbors. Moreover, we found that even when the amount of decisions made with our complementary module is small, this might have a significant impact on the degree of fairness in the energy consumed by the sensors.

Overall, irrespective of the particularities of the learning automata used, this paper shows that a significant part of the complexity and tweaking involved during the design phase of the automata can be relaxed by simply using a straightforward and well-known technique in networking, such as a routing selector based on a compound metric, which can work in concert with the automata in a synergetic way.

References

[1] P. Nicopolitidis, G. I. Papadimitriou, A. S. Pomportsis, P. Sarigiannidis, and M. S. Obaidat, "Adaptive Wireless Networks using Learning Automata," IEEE Wireless Communications, Vol. 18, issue 2, pp. 75–81, April 2011.

[2] S. Misra, V. Tiwari, and M. Obaidat, "LACAS: Learning Automata-based Congestion Avoidance Scheme for Healthcare Wireless Sensor Networks," IEEE Journal of Selected Areas in Communications, Vol. 27, no. 4, pp. 466–479, May 2009.

[3] S. Misra, P. Venkata Krishna, and K. I. Abraham, "A simple learning automata-based solution for intrusion detection in wireless sensor networks," Wirel. Commun. Mob. Comput., Vol. 11, issue 3, John Wiley and Sons, March 2011.

[4] M. Esnaashari, and M. R. Meybodi, "A learning automata based scheduling solution to the dynamic point coverage problem in wireless sensor networks," Computer Networks, Elsevier, Vol. 54, no. 14, pp. 2410–2438, October 2010.

[5] S. Misra and P. D. Thomasinous "A simple, least-time, and energy-efficient routing protocol with one-level data aggregation for wireless sensor networks," Journal of Systems and Software, Elsevier, Vol. 83, no. 5, pp. 852–860, May 2010.

[6] M. M. A. Azim, "MAP: A Balanced Energy Consumption Routing Protocol for Wireless Sensor Networks," Journal of Information Processing Systems, Volume. 6, no. 3, 2010.

[7] S. S. Chiang, C. H. Huang, and K. C. Chang, "A Minimum Hop Routing Protocol for Home Security Systems Using Wireless Sensor Networks," IEEE Transactions on Consumer Electronics, Vol. 53, no. 4, pp. 1483–1489, 2007.

[8] X. Zeng, R. Bagrodia, and M. Gerla "Glomosim: A Library for Parallel Simulation of Large Scale Wireless Networks," in Proc. of the 12th Workshop on Parallel and Distributed Simulations, 1998.