Computer Communications 94 (2016) 46–56

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

LA-CWSN: A learning automata-based cognitive wireless sensor networks

S. Gheisari[a,*], M.R. Meybodi[b]

[a] Department of Computer, Science and Research Branch, Islamic Azad University, Tehran, Iran
[b] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 10 January 2016
Revised 10 July 2016
Accepted 12 July 2016
Available online 14 July 2016

Keywords:
Cognitive
Learning automata
Traffic control
Wireless sensor network

ABSTRACT

Cognitive networking deals with using cognition to the entire network protocol stack to achieve stack-wide, as well as network-wide performance goals; unlike cognitive radios that apply cognition only at the physical layer to overcome the problem of spectrum scarcity. Adding cognition to the existing Wireless Sensor Networks (WSNs) with a cognitive networking approach brings about many benefits. To the best of our knowledge, almost all the existing researches on the Cognitive Wireless Sensor Networks (CWSNs) have focused on spectrum allocation and interference reduction, which are related to the physical layer optimization. In this paper, an inference and learning model for CWSNs, named LA-CWSN, is proposed. This model uses learning automata to bring cognition to the entire network protocol stack, with the aim of providing end-to-end goal. Learning automata are assigned to the parameters of the important network protocols. Each automaton has a finite set of possible values of the corresponding parameter, and it tries to learn the best one, which maximize the network performance. Each node in the network has its own group of learning automata, which act independently, however all nodes receive the same feedbacks from the environment. To clarify the proposed model a traffic control scenario in WSN is considered. Using the network simulator ns-2.35, we test the proposed inference and learning model for traffic control in a WSN. The results show that learning automata approach works well to apply cognition in WSNs.

1. Introduction

Wireless Sensor Networks (WSNs), which consist of small and inexpensive communication nodes, are the results of recent advances in processing capability, memory capacity, and radio technology [1]. These networks can sense their environments and communicating with each other. They are also developed at a cost much lower than traditional wired sensor systems. However, the design of WSNs is facing many challenges; some of them are as follows. Since the power source of the sensor nodes is limited and un-chargeable, energy efficiency and life time maximization are two key design factors. Spectrum allocation is another important problem in WSNs, because of the limited bandwidth and the spectrum scarcity, which cause interference in WSNs. Another challenge, which causes an end-to-end delay, is the sensing duration time. Collision occurs in WSNs because of the hidden terminal problem, whereupon control and data packets may be lost; so, managing access to the common medium is another important feature in designing WSNs.

To overcome some of the above designing challenges in a WSN, a cognitive technology has been proposed in [2–7]; the proposed solution, which has used cognitive radio, has been a new approach to the spectrum allocation and utilization concept. Generally, a cognitive radio [8] applies cognition only at the physical layer to dynamically detect and use spectrum holes, focusing strictly on dynamic spectrum access; whereas, the objective of cognitive networking [9–12] is to apply cognition to the entire network protocol stack for reaching network-wide performance goals.

Cross-layer design is another approach, which has used to overpower the designing challenges in WSNs. The cross-layer design refers to an explicit violation of the reference layer architecture [10], and it usually focuses on merging and/or splitting of layers, to achieve a particular goal. Every cross-layer design proposal serves to highlight a specific shortcoming of the traditional protocol layering [13], not end-to-end network goals. In addition, the cross-layer design is based on an algorithmic approach and does not learn and adapt according to an end-to-end communication goal [12]. So, cross-layer design is just a way to apply cognition to the entire network protocol stack in cognitive networks.

* Corresponding author.
E-mail addresses: S.gheisari@srbiau.ac.ir (S. Gheisari), mmeybodi@aut.ac.ir (M.R. Meybodi).

http://dx.doi.org/10.1016/j.comcom.2016.07.006
0140-3664/

Overall, unlike the cognitive radio and the cross-layer design, cognitive networks consider the entire network protocol stack to create a self-organized, self-aware, self-control, self-adaptive, and in short, an intelligent network. In a Cognitive Wireless Sensor Network (CWSN), sensor nodes would have additional state called *sensing state* [14]; they sense the environment, and use their observation to tune the controllable parameters of different protocols of the network protocol stack, in order to adapt with the environment and satisfy end-to-end goals. It has been shown that the coexistence of such networks can significantly increase a WSN's performance [15]. Raising the network efficiency by selecting proper parameters, and carrying out cross-layer optimization on the resulting stack, is a key functionality of any CWSN. Moreover, by the adaptive changes in the parameters of the network stack protocol, different data rates can be achieved, which in turn can directly influence power consumption and network lifetime.

In this study, we have focused on designing a learning automata-based CWSN. In the proposed model, which named LA-CWSN, a group of learning automata is assigned to each node to implement the cognition in a distributed way, and the nodes self adjust dynamically. To tune the parameters of the network stack protocol, in each node a learning automaton is assigned to each effective and controllable parameter; so all learning automata together, configure the network, to achieve the maximum network-wide performance. After setting the parameters, network starts its work and each automaton receives a reinforcement signal, which reflects the affect of the adjusted controllable parameter on the performance of the network. The reinforcement signal is one or more observable parameters[1] in the network. According to the received reinforcement signal, the automaton updates its probability vector. Gradually, each automaton learns the best value of the corresponding parameter, and consequently, all automata learn the configuration of the network protocol stack, which satisfies the network-wide performance metrics. To the best of our knowledge this is the first study, which proposes a practical solution for designing the CWSN. The proposed solution is neither a cognitive radio approach nor a cross-layer design; it is completely a cognitive network approach, in which the whole network protocol stack is considered to create a self-organized, self-aware, self-control, and self-adaptive network. The solution also takes advantage of the benefits of learning automata such as learning capability and low computational cost.

The rest of the paper is structured as follows. In Section 2 related works are briefly introduced. Learning automata is described in Section 3. The proposed model is explained in Section 4. Traffic control scenario in WSNs and the cognitive solution are presented in Section 5. In Section 6 performance evaluation is reported and the experimental results are shown in the form of charts and tables. Finally we conclude the paper in Section 7.

2. Related works

Since cognitive networks were introduced by Thomas in [12], much research has been done in the context of adding cognition to different networks such as, wireless networks, wireless mesh networks, wireless sensor networks, and etc. In this section, we survey the existing research works on cognitive wireless sensor networks (CWSNs). Research in the field of CWSNs can be classified into two main approaches; the first approach has studied the concept of CWSNs and some requirements for implementing them [14–20], while the second has studied specific issues such as routing, security, or channel assignment in CWSNs [21–24]. In this paper we have focused on the former and proposed a novel architecture.

In the first class of approaches, Zahmati et al. [14] have presented an overview of CWSNs; emerging topics, main advantages, and recent challenges in this area have been discussed, and also possible remedies to overcome the challenges have been suggested. Meshkova et al. in [16] have discussed the design of cross-layer optimization algorithm for cognitive wireless networks. They took a "black box" approach and studied the use of simulated annealing to maximize the network performance. To improve the convergence rate of the basic algorithm they have also applied graphical models on the perceived relations between network parameter and network utilities. Finally, they tested the proposed algorithm in simple WSNs. In [17] the concept of CWSNs has been reviewed and a preliminary case study has been illustrated. Vijay et al. [15] have provided the vision and the advantage of a holistic approach to add cognition in sensor networks, which can be achieved by incorporating learning and reasoning in the upper layer, and opportunistic spectrum access at the physical layer; the paper provides the reader with a framework based on knowledge and cognition that help to achieve end-to-end goals of application-specific sensor networks. In [18] a novel multilevel architecture for CWSNs has been proposed; the architecture has consisted of a Forest of Distributed Minimum Spanning Trees. Rovcanin et al. in [19] have analyzed and proposed a conceptual solution to determine which network services would be beneficial to enable cooperation in CWSN; a reinforcement learning technique known as the Least Square Policy Iteration has been used, and a self-learning entity, which negotiates between different independent and co-located networks has been proposed. In [20] a versatile platform has been presented that brings cognitive properties into WSNs; it has combined hardware and software modules as an entire instrument to investigate CWSN.

The mentioned works have focused on foundations of CWSNs, such as network architecture; however, some more specialized works, which place in the second class, have also been done as the following. In [21] a distributed leveling and clustering-based quality of service routing protocol has been presented, which can increase the network lifetime in CWSN by saving some battery power. In [22,23] security and privacy challenges in CWSNs have been mentioned and various security threats and defense mechanisms have been discussed. In [24] the problem of robust topology control in CWSNs with the purpose of assigning a channel to each radio such that the resulting topology is robust to the presence of a primary user [8] has been addressed and also a distributed algorithm for channel assignment has been proposed.

As mentioned previously, in this paper a novel learning automata-based architecture for CWSN has been proposed. However Meshkova et al. [16]. have proposed a cross-layer optimization rather than a cognitive network, their model is the most similar one to our proposed model named LA-CWSN. Some of the main differences between LA-CWSN and their model are: using learning automata rather than simulated annealing; having dynamic self-adjusting in each node, which separates the cognition concept from the cross-layer design; and ignoring any added graphical model to show the relationship between the parameters.

3. Learning automata

In this section, learning automata [25,26] will be briefly reviewed.

A learning automaton is an adaptive decision-making device that enhances its functionality through learning how to choose the optimal action from a finite set of permitted actions by repeated interactions with a random environment. In this case, the environment evaluates this taken action and responds by a reinforcement signal. The learning automaton then updates its internal information, which is in the form of a probability vector, according to both

[1] In some references observable parameters are known as the attributes of the network.

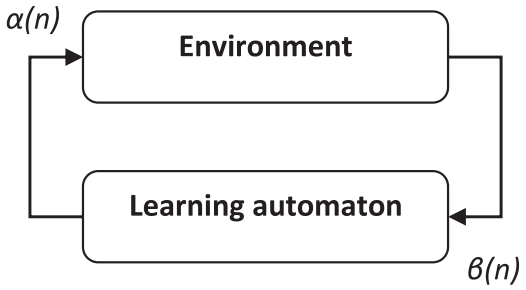

Fig. 1. The interaction between learning automaton and the environment.

the chosen action and the received reinforcement signal. Then, the learning automaton adjusts its action repeatedly, until a termination condition is satisfied. Fig. 1 represents the interaction between a learning automaton and its environment.

Every environment is indicated by $E = \{\alpha,\ \beta,\ c\}$, where $\alpha = \{\alpha_1, \alpha_2, \ldots,\ \alpha_r\}$ is the set of inputs, $\beta = \{\beta_1,\ \beta_2, \ldots,\ \beta_r\}$ is the set of outputs, and $c = \{c_1, c_2, \ldots,\ c_r\}$ is the set of penalty probabilities. Depending on the nature of the reinforcement signal β, the environments are classified into P-model, Q-model, and S-model. Whenever the set β has just two members, $\beta_1 = 1$ and $\beta_2 = 0$, the environment belongs to the class of P-model. In the Q-model class, the environment contains a finite number of values in the interval [0,1]. In contrast, the S-model environment has an infinite number of members. Finally, c_i denotes the penalty probability of each taken action α_i.

Learning automaton is categorized into fixed structure and variable structure. Learning automaton with variable structure, which will be used in this paper, is represented by $\{\alpha, \beta, p, T\}$, where $\alpha = \{\alpha_1, \alpha_2, \ldots,\ \alpha_r\}$ is the finite set of actions, $\beta = \{\beta_1,\ \beta_2, \ldots,\ \beta_r\}$ is the set of inputs, $p = \{p_1,\ p_2, \ldots,\ p_r\}$ is the action probability vector, and $p(n+1) = T\ [\alpha(n),\ \beta(n),\ p(n)]$ is the learning algorithm. After choosing an action, α_i, the learning automaton, upon receipt of a reinforcement signal from the environment, updates its action probability vector according to Eq. (1).

$$
\begin{aligned}
&p_i(n+1) = p_i(n) + a[1 - p_i(n)],\\
&p_j(n+1) = p_j(n) - a p_j(n)\ \forall j,\\
&\quad j \neq i \quad \text{If the received signal is taken as desirable}\\
&p_i(n+1) = (1-b)p_i(n),\\
&p_j(n+1) = \frac{b}{r-1} + \left[(1-b)\ p_j(n)\right] \forall j,\\
&\quad j \neq i \quad \text{If the received signal is taken as undesirable.}
\end{aligned}
\tag{1}
$$

Where, $p_i(.)$ is the probability of the selected action, and $p_j(.)$ defines the probability of other actions in the action-set α; r denotes the number of actions in the action-set α; and finally, the parameters a and b denote reward and penalty parameters, respectively. If $a=b$, the algorithm is called $L_{R\text{-}P}$.; if $b=0$, the automaton is called $L_{R\text{-}I}$, and if $b=\varepsilon a$, with $\varepsilon < 1$ being a small number the automaton is called $L_{R\varepsilon P}$ [26]. More information about learning automata can be found in [25,26].

For a more complete description of learning automata, a single automaton is generally sufficient for learning the optimal value of one parameter. But for multidimensional optimization problems, we need a system consisting of as many automata as there are parameters [25]. Let $A_1, \ldots, A_N$ be the automata involved in an N-player game. Each play of game consists of each of the automaton choosing an action and then getting the *payoffs* or reinforcements from the environment for this choice of actions by the group of learning automata. Let $p_1(n),\ p_2(n), \ldots,\ p_N(n)$ be the action probability distributions of N automata. Then at each instant n, each automaton A_i chooses an action $\alpha^i(n)$ independently and at random, according to $p_i(n), 1 \leq i \leq N$. This set of N actions is input to the environment, which responds with N random payoffs. Random payoffs are supplied as reinforcements to the corresponding automaton. Special case of this model is a game with *common payoff*; here, all the automata get the same payoff from the environment (that is, $\beta^i = \beta$). Learning automata in a common payoff game is often referred as a team of learning automata.

4. Proposed model for CWSNs

In this section, the problem of designing LA-CWSN, which is a learning model for CWSN, has been described and our suggested model has been presented. Then, some assumptions have been discussed and a formal description has been expressed. Finally, the major properties of LA-CWSN and the implementation requirements have been given.

4.1. Problem formulation

As already mentioned implicitly, the parameters of the network protocol stack are classified to controllable and observable parameters; and observable parameters, consisting application-level parameters, are affected by controllable parameters. Some examples of controllable parameters in a CWSN are: the duration of sensing interval in the application level, the value of congestion window in the transport protocol, and the maximum number of transmission in the MAC protocol. Similarly, some examples of observable parameters are: power consumption, Round Trip Time (RTT) in the transport protocol, and the number of retransmission in the MAC protocol. Such representation of parameters allows for a simple control loop, in which controllable parameters are tuned in accord with observable parameters to improve the performance of the network. The quality of the observable parameters can be measured individually or using an application specific utility function [27]. In the former, each controllable parameter associates with one or more observable parameters and directly uses their values, to tune itself. In the later, utility functions are used, as a flexible way, to define the satisfaction; they allow for a quantifiable expression of user needs [28] and they are used by controllable parameters to tune themselves. The common forms of utilities are linear, logarithmic and step-wise functions, with components that can be additive, weighted additive, or raised to power of a weight. However, other forms of objective functions can be used as well.

In this study, the controllable parameters adjustments are conducted by learning automata; each controllable parameter also associates with one or more observable parameters, whose qualities are measured individually and used by learning automata to tune the controllable parameters. The overall proposed network model is represented in Fig. 2.

4.2. Assumption

WSNs and also CWSNs are characterized by their applications or objectives that their users want to achieve. Network parameters are optimized at different time scales; for example, switching between sensor states takes longer than adjustment of a single MAC parameter. We consider this fact, and update the interior state of each learning automaton in the right time. Also, to have a more effective learning, the network-wide performance is also measured at specific times, by using a utility function.

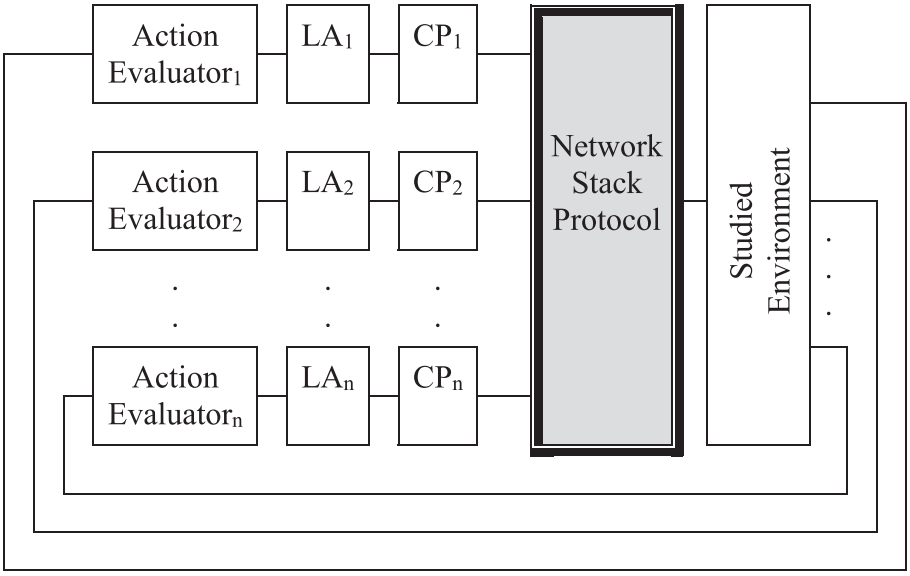

Fig. 2. The proposed model of CWSN.

4.3. Formal description[2]

In LA-CWSN, each group of learning automata, corresponding with a node, searches among all permutations of controllable parameters values of the network protocol stack to maximize the network-wide performance. The formal description of the problem is as follows.

Let S be the collection of all possible configurations of the network protocol stack in a node; each configuration is subject to several constraints, c: $S \rightarrow \{0, 1\}$, such as, protocols that are mutually exclusive, or protocols, which are a subset of each other and do not add to the overall functionality, or users' constraints. The set of configurations, which satisfy the constraints, is simply defined by, $S_c = \{x \in S | c(x) = 1\}$.

Let $CP(s)$ be the list of controllable parameters in the network protocol stack. Controllable parameters themselves, may also be subject to further constrains, $c'_s : CP(s) \rightarrow \{0, 1\}$, which are either globally delimited, or are a result of a combination of several protocols in the network protocol stack. We can therefore derive a set of valid controllable parameter value vectors that are applicable to the stack, which is given by, $CP_{c'_s} = \{x \in CP | c'_s = 1,\ s \in S\}$.

The performance of the applied network protocol stack is described by a set of observable parameters OP. The OP is not always defined only from a technical point of view. Economic considerations, such as the cost of deploying a certain topology, bandwidth-based fees and other application-specific parameters may also be taken into account. According to Fig. 2, we classify the observable parameters corresponding with each controllable parameter, and then evaluate their values individually to feed the controllable parameters. In this situation, the desirable network protocol stack configuration is described as the result of separated evaluations. However, as mentioned in Section 4.2, to evaluate the network-wide performance periodically, a utility function is also used. The utility function, $U(OP(s, cp, E))$, maps the observable parameters' values to a single numerical, according to the deployed stack ($s \in S, cp \in CP$), and the operational environment E. Higher values of the utility function show the better adjusted controllable parameters' values, over the other. Finally, finding the maximum value of the utility function, which is represented in Eq. (2), is the ultimate goal of the proposed model.

$$(s, cp) = \underset{(s,cp) \in \left(S_c \times CP_{c'_{S_c}}\right)}{\operatorname{argmax}} U(OP(s, cp, E)) \tag{2}$$

4.4. The properties of LA-CWSN

LA-CWSN has several properties, which distinguish it from classical pre-planning in WSNs. First, it does not need any definite starting point, which is a default setting of the controllable parameters of the network protocol stack. In the beginning, all possible values of controllable parameters have the same chance to be chosen and in each repetition, the probability vectors of the learning automata update according to the received reinforcement signals from the network environment. This property helps to prevent of sticking in a local optima; however, it may cause execution time overhead.

Moreover, LA-CWSN is carried out in two phases: offline learning phase, and inference and online learning phase. The offline stage is regarded as a long-term learning, i.e. the performance of the network is monitored for a long time without any need to keep high average reward, during the learning period. This stage can take place at times, when the network load is low; for example, during the night times. The offline learning phase leads to better network protocol stack configurations; however, for all networks is not possible to have such stage. The online phase has more constraints. First, it must have shorter learning periods, not to disturb the user. Second, reinforcement values must be kept high for satisfying users. For the same reason, the network must react fast to the changes of the network conditions, which can be stated as inferring in a cognitive network. Ideally, in a cognitive network we have both of these phases.

Finally, sudden changes in the utility function value may happen due to a temporary and unimportant event, and the network protocol stack configuration should not be affected by such changes. However, for an exception, which is a drastic change in the network's performance, LA-CWSN should recognize the change, and react immediately. As an example for the later, there might be a sudden raise in the node failure rate when the network shifts from low-power message delivery to just message delivery. Distinguishing these two cases depends on the time limitations in

[2] In this section we have widely used the reference [16] formulations.

different applications; and in a LA-CWSN, the network gradually learns to react automatically and appropriately in both situations.

4.5. The implementation

In this section, we describe the proposed learning automata-based model for adding cognition to the WSN. A pre-defined threshold, max_{it} is determined, and the learning phase is repeated up to max_{it} times. After that, the search procedure stops and the parameters values, which have yielded the highest utility, are returned. So, each learning automaton should have a more memory to keep the value, in which the network has had the best performance.

In the learning phase, each learning automaton, which has been assigned to a controllable parameter, chooses an action from its action sets. The action sets of learning automata contain the possible values of their corresponding controllable parameters. As mentioned in Section 3, action sets are finite and if any continuous parameter exists, it should be quantized. Using the chosen values, the parameters are set and therefore, the network protocol stack is configured. The configured network then interacts with the environment and does its functionalities; based on this interaction the observable parameters are affected. Learning automata get their own feedbacks from the environment according to the corresponding observable parameters. Whenever an automaton receives the feedback, it individually evaluates that it is a desirable or undesirable one. For a desirable value of the observable parameter an automaton rewards its chosen action and for an undesirable one it penalizes the action. All automata act autonomously and in a distributed form. Progressively, the probability vectors of the actions in all automata will converge and they can choose the best values for the controllable parameters; so the whole network will be configured properly to adapt with the environment situation.

In order to overcome temporary and unimportant variations in utility (basically background noise because of the changing utilization of the network), we extended the LA-CWSN by a system of comparing the values of two sliding windows, with window sizes adjustable to the variance in the utility.

Since the learning is usually applied at runtime, we have further tried to limit the negative impact of the online learning to the users. As the network (especially during early stages) tries to find the parameters combination that yields high utility, we add several conditional barriers; so, we would be able to cope with high fluctuations in the utility. For example, once the LA-CWSN converges to a good stable solution, the learning is stopped; however, the observable parameters and the utility values are continuously monitored over the time. If the system notices a sudden drop in the performance that cannot be traced back to a temporary and unimportant variance, but indicates a change in the network usage, learning phase is restarted with agility. A static memory of the best parameter values, discovered over time, is used to boost up the algorithm performance by letting it first try those combinations.

We are also able to make a tradeoff between the duration of the learning period and the average utility achieved during the learning period. The learning iterations may be conducted only every N^{th} iteration, for all other returning to the best state. This allows limiting the negative experience for users at the expense of a prolonged convergence time.

5. Traffic control scenario in LA-CWSN

WSNs and also CWSNs have important features, which affect the network performance; some of them are listed below. They have low-power radios because the power sources of the sensor nodes are limited and un-chargeable. Their performance is extremely sensitive to the deployment environment, because the signals of the low-power radios are easily disrupted. There are no widely accepted well-performing protocol stacks for these networks, and several protocols and many parameters might be considered for deployment.

Traffic control generally refers to set policies and mechanisms that allow a network to efficiently satisfy a diverse range of service requests; traffic control is also needed for providing quality of service. According to the mentioned features of WSNs and CWSNs, two basic challenges exist for traffic control [29]: efficient resource utilization and quality of service guarantee.

In this section, first, we determine the traffic control parameters and then, we examine how learning automata will be able to adjust these performance.

5.1. Traffic control parameters

There are many parameters in a WSN protocol stack, which should be considered in order to efficiently control the traffic. Here, we have chosen seven controllable parameters: Sensing Interval (SI) in the application layer, which determines the duration of asleep and awake; Congestion Window (CW) in the transport layer, i.e., the maximum number of packets that can remain unacknowledged at any time; Beaconing Interval (BI) in the network layer, which defines routing duration in a routing protocol, i.e. a maximum number of slots a node wait after sending rout request packet; and in the MAC layer, Duty-Cycle (Du.C), i.e. the maximum number of slots a node waits to receive the acknowledgment of a sent packet and then retransmit the packet; the Maximum number of Retransmission (Mx.R), which limits the retransmission attempts after an unsuccessfully packet transmission; Contention Window (Ct.W), i.e. the maximum number of slots a node wait before sending a packet according to a random back off interval between zero and contention window; and Acknowledgement-enable (ACK), a binary variable, which determines if a node must wait for the acknowledgement after sending a single packet or not.

To examine the effectiveness of a selected network protocol stack configuration, nodes must be able to observe the feedback of the environment; for this reason, some observable parameters are considered, which can be listed as below. Event Duration (ED) in the application layer, i.e. duration of time that the nodes of the network must be awake and sense the environment, corresponding with the event occurred in the environment. In the transport layer, Congestion Status (CS), which is a binary variable to define the occurrence of the congestion; Round Trip Time (RTT), which is the time between when a packet is sent and when the corresponding acknowledgment is received; Throughput (Thp), i.e. the total amount of unique data [bytes] acknowledged in a sampling interval. Rout Request Ack (RRA), in the network layer, a binary variable, which specifies that a rout is found. In the MAC layer, Packet Loss (PL), which is a binary variable and defines either the acknowledgment of a sent packet has been received or not; Total Transmission (TT), i.e. the total number of original MAC packets sent in the sampling interval; Total Retransmission (TR), which is the total number of MAC retransmissions in the sampling interval; finally, Power Consumption.

Since we want to evaluate the performance of the algorithm on scenarios, where an exhaustive search is possible, total number of controllable parameters values permutations have to be limited, and also each parameter must have a finite set of the values to be adjusted. Most of the above parameters have a finite number of outcomes; also some of them, such as RTT, have been quantized to n_q levels, so, all variables are multinomial, which is computationally more efficient.

Table 1
Defining the parameter used in flow the flow chart of traffic control in LA-CWSN.

Abbreviation	Definition
SI	Sensing Interval: duration of asleep and awake and is set by learning automaton LA1
BI	Beaconing Interval: routing duration in a routing protocol and is set by learning automaton LA2
CW	Congestion Window: the maximum number of packets that can remain unacknowledged at any time and is set by learning automaton LA3
ACK	Acknowledgment: a binary variable, which determines if a node must wait for the acknowledgement after sending a single packet or not and is set by learning automaton LA4
Du.C	Duty cycle: the maximum number of slots a node wait to receive the acknowledgment of a sent packet and then retransmit the packet and is set by learning automaton LA5
Mx.R	Maximum number of Retransmission: the number of retransmission attempts after an unsuccessfully packet transmission and is set by learning automaton LA6
Ct.W	Contention Window: the maximum number of slots a node wait before sending a packet according to a random back off interval between zero and contention window and is set by learning automaton LA7
RRA	Rout Request Acknowledgment: a binary variable, which specifies that a rout is found
CS	Congestion Status: a binary variable to define the occurrence of the congestion
PL	Packet Loss: a binary variable and defines either the acknowledgment of a sent packet has been received or not
ED	Event Duration: duration of time that the nodes of the network must be awake and sense the environment, corresponding with the event occurred in the environment

5.2. Learning automata settings

After determining the parameters and the finite sets of their values, a learning automaton is assigned to each controllable parameter in each node. Each learning automaton has the following characteristics:

Set of actions (α): the finite set of possible values of each controllable parameter.

Output (β): the output or feedback of each automaton is computed based on the value of its corresponding observable parameter(s) value(s); for example, for the learning automaton, which is assigned to BI the value of RRA will be checked; for the learning automaton, which is assigned to CW the value of CS will be checked; for the learning automaton, which is assigned to Du.C the value of PL will be checked, and …. If the value of the observable parameter, which is usually a binary value, is desirable the corresponding automaton will reward the selected action and otherwise it will penalize it (a P-model environment is considered).

Probability vector (p): initially, we use the uniform probability distribution, and as the algorithm runs, the probability vectors of learning automata will be updated using a learning function.

Learning function (T): linear reward penalty algorithm presented in Section 3 is used.

In each repetition, each automaton chooses an action from its action set; actions chosen by all automata configure the network protocol stack. Then the values of the observable parameters are used to update the probability vectors of the learning automata. All steps should be repeated for max_{it} iterations and the parameter values, which yielded the highest utility, are returned. Fig. 3 depicts the flow chart of traffic control in LA-CWSN and Table 1 summarized all abbreviations to clarify the figure.

As it is shown in Fig. 3, the learning automata, named LA1 is assigned to SI; likewise, LA2 to BI, LA3 to CW, LA4 to ACK, LA5 to Du.C, LA6 to Mx.R, and LA7 to Ct.W.

Moreover, to evaluate the network-wide performance periodically, an application specific utility function, which allows for a quantifiable expression of user needs, is used. In this paper, different utility functions can be used; the utilities are functions of observable parameters' values, such as throughput, delay, reliability (maximum number of packet lost), and also power consumption. As an example we use the utility as a function of power consumption with a constraint on the reliability; because in a traffic control scenario in WSN energy and the number of packets lost are two key factors. Higher values of the utility function show the better adjusted controllable parameters' values, over the other. After finishing the offline learning phase (if having such phase is possible), LA-CWSN always monitors the output of the utility function to evaluate the performance of the configured network protocol stack, and if a permanent or a drastic decrease in the utility is observed, learning phase restarts.

Table 2
controllable parameters and their values.

Protocol	Parameter	Values
Application	SI	200, 800, 1400
Transport	CW	1, 3, 5
Routing	BI	5, 20, 40
MAC	Du.C	10, 50, 80, 100
	Mx.R	1, 3, 9
	Ct.W	5, 30, 40
	ACK	"on", "off"

6. Performance evaluation

LA-CWSN is implemented in NS-2.35 [30] and its performance is evaluated over a wide range of scenarios. Network size is varied from fifty to four hundred nodes, with random topologies. Network is queried every n milliseconds and information on the nodes performances is gathered, and parameters of the nodes are adjusted as the result of learning process. As previously mentioned, to evaluate the performance of the algorithm on scenarios, where the exhausted search is possible, we have to limit total number of parameters values permutations. Table 2 shows the possible controllable parameters values.

We classify our experiments into two categories: experiments to study the effect of algorithms and parameters of learning automata (please refer to Section 3), and also comparative experiments, in which LA-CWSN is compared against the WSN without cognition and other proposed model in [16], which uses simulated annealing and we will name it SA-CWSN.

6.1. Study the effect of learning algorithms and parameters

In this section, we study the effect of the learning algorithms and the learning parameters on the performance of LA-CWSN. As mentioned in Section 3, there are two learning parameters a and b in Eq. (1), and also three learning algorithms, $L_{R\text{-}P}$, $L_{R\varepsilon P}$, and $L_{R\text{-}I}$ based on the values of learning parameters. The main differences between $L_{R\text{-}P}$ and $L_{R\text{-}I}$ is that, under $L_{R\text{-}I}$ when the reinforcement signal is undesirable the action probability vector is left unchanged. It has been shown [26] that in the case of the $L_{R\text{-}I}$ algorithm, the action probability vector converges to a unit vector. Using the $L_{R\text{-}P}$ algorithm, which always updates the probability vector, the unit vector cannot be a stable point of the

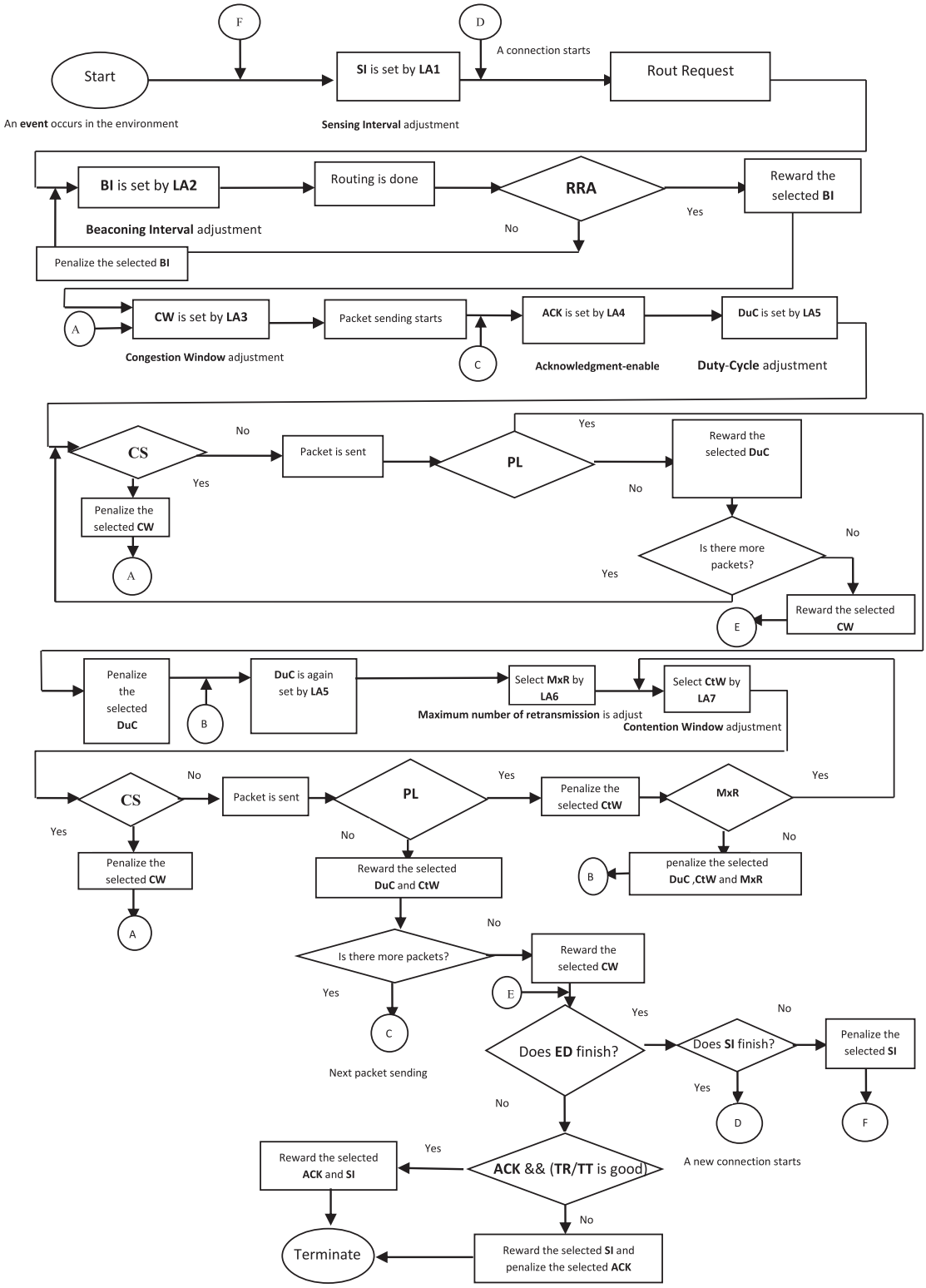

Fig. 3. Flow-chart of traffic control in LA-CWSN.

process $p(n)$. It is reasonable, because there is always a chance that the probability of the selected action is decreased and the action probabilities converge in distribution. In this situation, if the learning parameter $(a=b)$ is sufficiently small, then the limiting value of p_i, probability of i^{th} action, would essentially be proportional to $\frac{1}{1-d_i}$, where, d_i is the reward probability of i^{th} action [31]. Compared to $L_{R\text{-}I}$, using $L_{R\text{-}P}$ algorithm learning automata can response to environment changes faster [26] and [31]. The properties of $L_{R\varepsilon P}$ are between those of $L_{R\text{-}P}$ and $L_{R\text{-}I}$. Generally, we can conclude that, in a dynamic environment, such as LA-CWSN, using $L_{R\text{-}P}$ causes the values of controllable parameters are updated more quickly when network conditions have changed; therefore, $L_{R\text{-}P}$ has been chosen as the main learning algorithm in LA-CWSN.

Table 3
The effect of learning parameters under L_{R-P} learning algorithm.

Learning parameters (a=b) Performance metrics	0.0001	0.001	0.01	0.1	0.2	0.3	0.5	0.7	0.99
RTT (delay)	1.002	0.902	0.715	0.547	0.671	0.707	0.750	0.803	0.957
Throughput	75.892	82.225	90.012	95.978	92.992	90.525	89.645	85.078	80.321
Packet loss	750.5	598.0	537.2	518.1	525.12	530.0	545.5	592.5	600.1
Power consumption	20.078	17.001	8.593	8.237	8.379	8.503	9.031	10.017	17.123
Utility	0.7215	0.8151	0.8975	0.9740	0.9450	0.8813	0.8228	0.7730	0.7002

Table 4
The effect of different learning algorithms.

Learning parameters (a=b)	LR-P a=b=0.1	LRεP a=0.1, b=0.01	LRεP a=0.1, b=0.001	LR-I a=0.1, b=0
Performance metrics				
RTT (delay)	0.547	0.701	0.713	0.721
Throughput	95.978	90.222	90.075	89.165
Packet loss	518.1	591.3	595.5	599.0
Power consumption	8.237	9.071	9.219	9.563
Utility	0.9740	0.9250	0.9119	0.9031

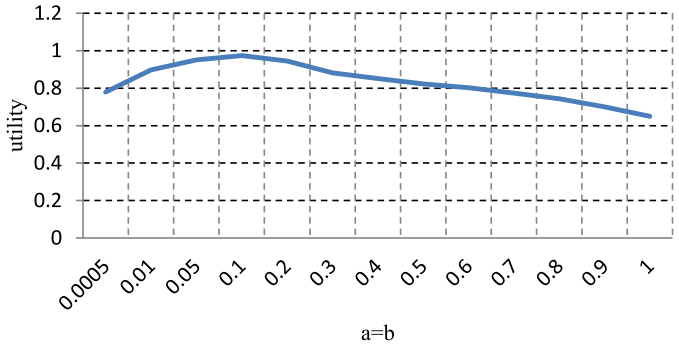

Fig. 4. The effect of learning parameters, under L_{R-P} learning algorithm, on the performance of LA-CWSN based on the utility value.

In the following, firstly, the effect of different values of the learning parameters has been studied and then, a brief comparison between different learning algorithms has been done. The simulation has been carried out in 400 × 400 area, with 200 sensor nodes, which have been set up at random. The number of iterations has been set to 80.

First simulation, whose results are given in Table 3 and Fig. 4, has been conducted in order to study the effect of learning parameters a and b, under L_{R-P} learning algorithm, on the performance of the LA-CWSN.

From the reported results in Table 3 and also Fig. 4, we can conclude that, the best results are obtained with $a=b=0.1$. As it can be seen, smaller and larger values show worst performance based on utility and also other metrics. This is justifiable because increasing the value of learning parameter causes the probability vectors constantly change, and on the other hand, by decreasing the value of learning parameter the speed of learning automata reduces and they achieve the desirable values very slowly.

In the next simulation, the effect of different learning algorithms L_{R-P}, $L_{R\varepsilon P}$, and L_{R-I} on the performance of the LA-CWSN, is studied. Based on the previous results, $a=0.1$ and under the $L_{R\varepsilon P}$ algorithm $b=0.1a$ and $b=0.01a$. The results are given in Table 4.

Reported results in Table 4 depict that by choosing the smaller values of ε, the results will approach to the results of L_{R-I} algorithm; anyway L_{R-P} shows the best results based on the utility value, in comparison with other learning algorithm. Thus, next simulations are done using L_{R-P}, as the learning algorithm, and $a=b=0.1$ as the learning parameters.

6.2. Compare the LA-CWSN against simple WSN and SA-CWSN

In the following, the performance of the LA-CWSN is evaluated over a wide range of scenarios using different performance measures and network sizes. All network scenarios are carried out in NS-2.35. Fig. 5 illustrates the results. Experiments are repeated 50 times, and the obtained results are averaged. Performance measures are power consumption depicted in Fig. 5(a), average delay shown in Fig. 5(b), total number of packet loss represented in Fig. 5(c), total amount of unique data, which are acknowledged as throughput, illustrated in Fig. 5(d), total number of delivered packet depicted in Fig. 5(e), and finally, the utility as a function of power consumption with a constraint on the number of packet loss shown in Fig. 5(f). Network size varies from 50 to 400 nodes with random topology.

As it can be seen in the charts, CWSNs usually have much better performance in comparison with WSN without cognition; it was predictable, since CWSNs tune the parameters to improve the performance, whereas a WSN without cognition adjusts the parameters with no attention to the environment conditions. On the other hand, compared with SA-CWSN, LA-CWSN shows better performance. LA-CWSN is always superior to other networks based on the performance measures; while SA-CWSN, which takes advantage of simulated annealing and also simple graphical models, sometime shows nearly the same performance with simple WSN. Moreover, SA-CWSN, actually, is a cross layer optimization design, which has a centralized view to tune the network protocol stack; i.e., all nodes use the same parameters' values and all nodes evaluate their parameters in the same way; though, the main advantage of "cognition" over "pre-planning" is the fact that the nodes self-adjust dynamically. In contrast, LA-CWSN is quite a cognitive approach, and nodes tune their protocol stacks in a distributed and autonomous way. Also, using learning automata has more profits in comparison with using simulated annealing; for example, learning automata, which have learning capability, test more states in the search-space and are less likely to remain in the local optima.

In next experiments, the effect of the number of iterations on the performance of the algorithms is studied. In these simulations we examine LA-CWSN in two cases: LA-CWSN1, which has been discussed till now, and LA-CWSN2, in which all learning automata, LA1 to LA7, receive the same feedback from the network environment; this identical feedback is the utility value as discussed before. Fig. 6 shows the results. Here, too, the experiments are repeated 50 times, and the obtained results are averaged.

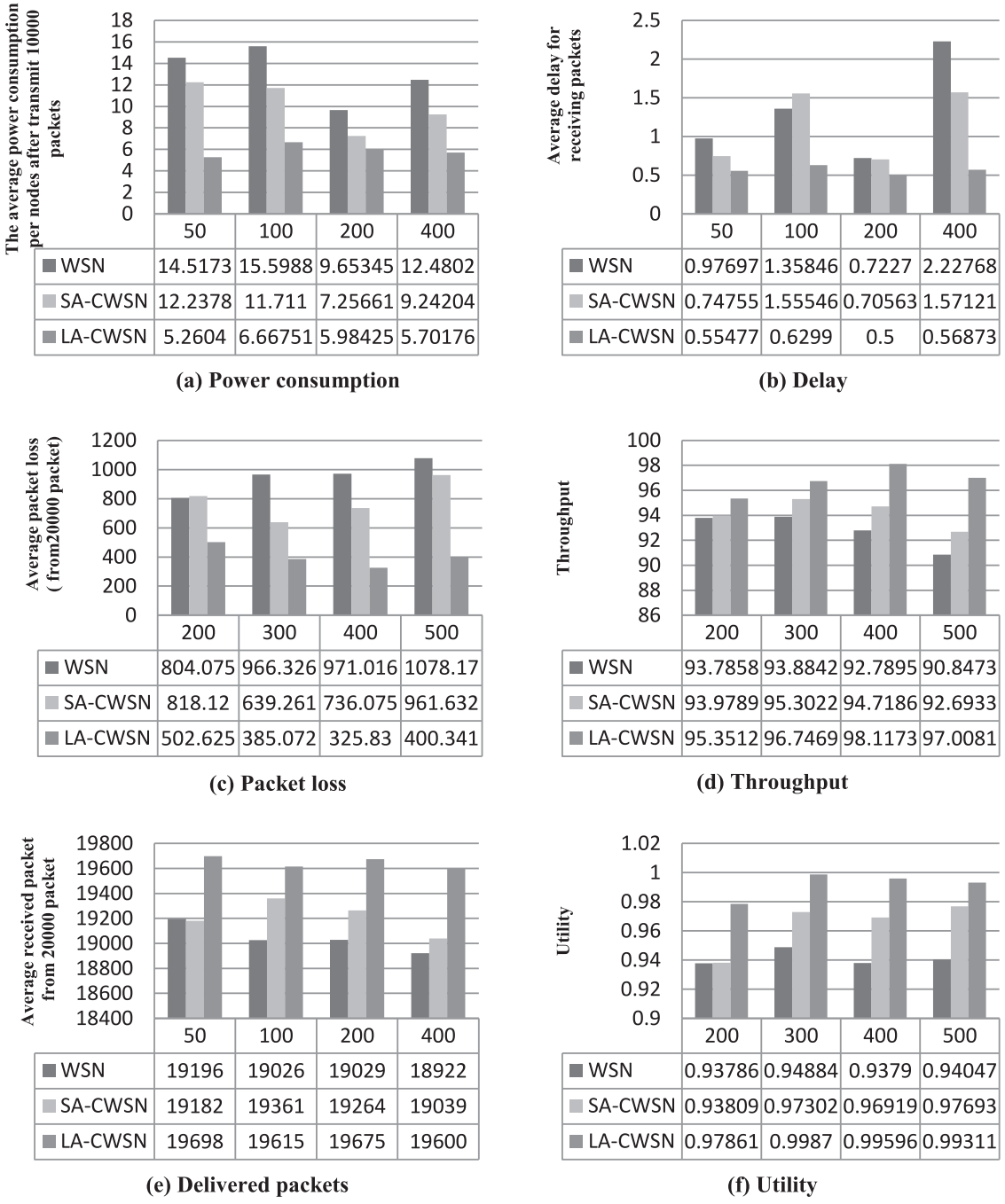

Fig. 5. Comparison the performance of LA-CWSN with SA-CWSN and the basic WSN (the charts are obtained by conducting several experiments at random starting parameter configurations and then averaging over these results is done.)

As the results indicate, both LA- CWSNs converge to their best performance faster than SA- CWSN. Also, comparing the LA-CWSN1 with LA-CWSN2 demonstrates that the basic LA-CWSN, in which learning automata receive different payoffs from the network environment, has better results and it converges to its best results faster. LA-CWSN1 converges to the best results almost always in iteration 70, and it converges to the 80% from maximum after about 50–60 iterations; while the best results for LA-CWSN2 are usually reached after iteration 75.

Fig. 6(a) depicts that, power consumption of LA-CWSN1 is gradually reduced and it can save more energy compared with LA-CWSN2 and specially SA-CWSN. Similarly, Fig. 6(b) for average delay, Fig. 6(c) for total number of packet loss, Fig. 6(d) for throughput, and Fig. 6(e) for utility illustrate that LA-CWSN1 is superior to the both LA-CWSN2 and SA-CWSN

However it seems that LA-CWSN has been tested only in one scenario, traffic control is a comprehensive scenario, in which many controllable parameters must be considered; so, we can claim that LA-CWSN works well in other scenarios of WSNs, which are often more simple. As an example target tracking [32–34], in which predicting the target position and awakening right sensor nodes are two main problems, can be mentioned. Topology control [35–37], in which the positions of nodes and the connectivity of the network are two important issues, is another example. Energy aware routing [38] is another instance that considers some parameters. Overall, adding cognition to the existing WSNs with

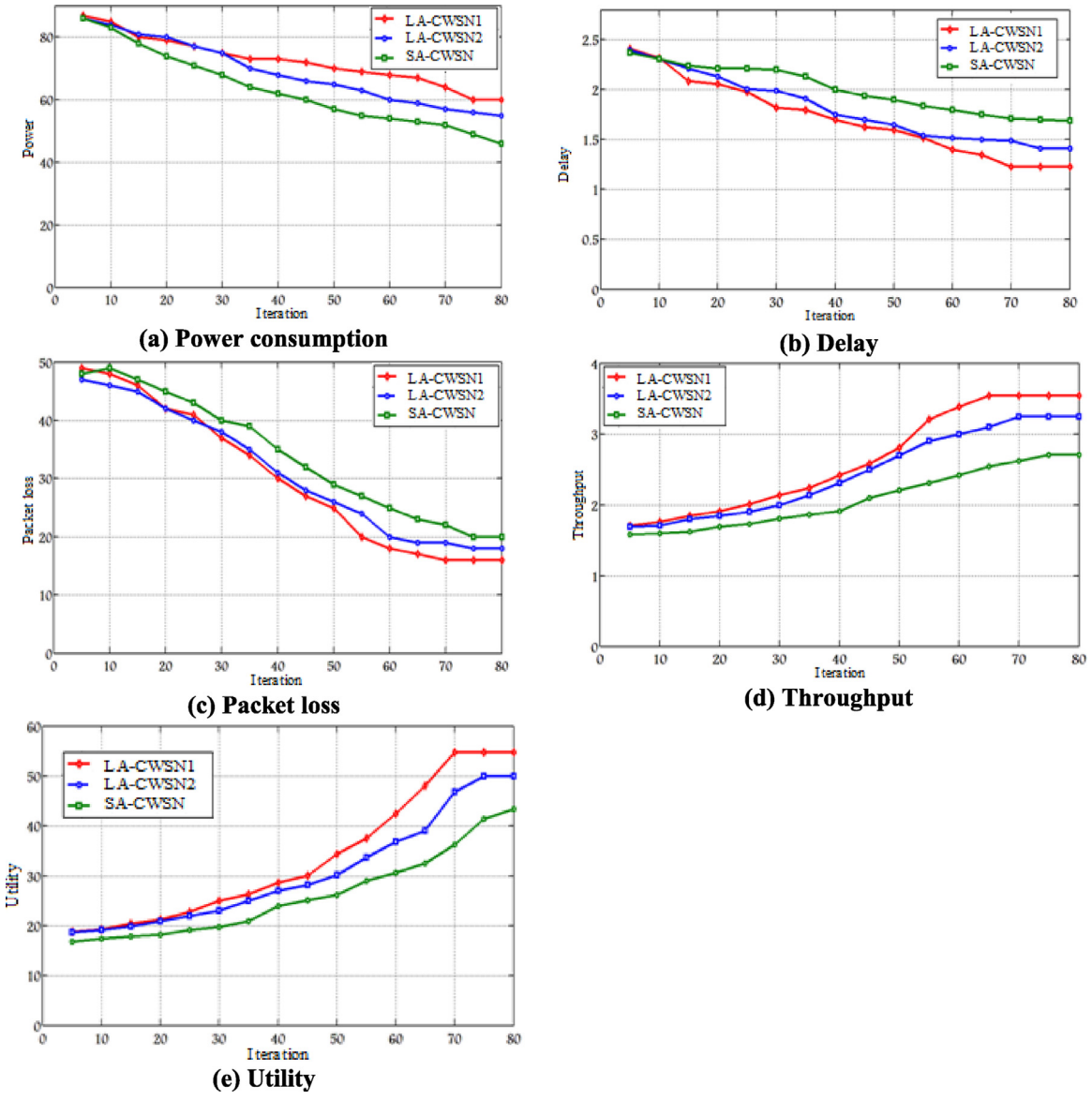

Fig. 6. The effect of the number of iterations on the performance metrics.

a cognitive networking approach brings about many benefits, and using learning automata, due to their learning capability and low computational cost, is an effective way to implement the cognition loop in CWSNs.

7. Conclusion

In this study, adding cognition to the common WSNs and the properties of the CWSNs have been discussed; then, a learning automata-based model, which called LA-CWSN, has been proposed. In LA-CWSN, learning automata have been used to tune the networks controllable parameters. Each learning automaton is assigned to one controllable parameter and it chooses one possible value to adjust the parameter. Each learning automaton receives its own payoff from the network environment, and updates its internal information using the payoff and the probability vector of the actions. Over time, learning automata learn the suitable values of the parameters, corresponding with the current situation of the network. All the nodes in the network have this set of learning automata to dynamically configure the network protocol stack by itself; indeed, LA-CWSN is a distributed and intelligent model to implement a CWSN. Finally, the proposed LA-CWSN has successfully been tested. The results have represented the superior results compared with a basic WSN and also the simulated annealing based CWSN.

Cognitive networks are popular within the areas of communication networks and AI. The authors yearn to continue their investigations in this field, by using other machine learning mechanisms such as Particle Swarm Optimization (PSO) and Bayesian Networks (BNs). In addition, other important scenarios in WSNs, such as target tracking, topology control, and energy aware routing are interesting research topics, which can be considered in CWSNs and will be the subject of the authors' next investigations.

References

[1] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330.
[2] D. Cavalcanti, S. Das, J. Wang, K. Challapali, Cognitive radio based wireless sensor networks, in: Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on, IEEE, 2008, pp. 1–6.
[3] S. Gao, L. Qian, D.R. Vaman, Distributed energy efficient spectrum access in wireless cognitive radio sensor networks, in: Wireless communications and networking conference, 2008. WCNC 2008. IEEE, IEEE, 2008, pp. 1442–1447.

[4] J. Ansari, X. Zhang, P. Mähönen, A decentralized MAC protocol for opportunistic spectrum access in cognitive wireless networks, Comput. Commun. 36 (13) (2013) 1399–1410.
[5] J. Wu, Y. Dai, Y. Zhao, Effective channel assignments in cognitive radio networks, Comput. Commun. 36 (4) (2013) 411–420.
[6] S. Althunibat, M. Di Renzo, F. Granelli, Cooperative spectrum sensing for cognitive radio networks under limited time constraints, Comput. Commun. 43 (2014) 55–63.
[7] J. Backens, C. Xin, M. Song, A novel protocol for transparent and simultaneous spectrum access between the secondary user and the primary user in cognitive radio networks, Comput. Commun. 69 (2015) 98–106.
[8] J. Mitola, CognitIVE RADIO: An Integrated Agent Architecture for Software Defined Radio PhD thesis, Royal Institute of Technology (KTH), 2000.
[9] R.W. Thomas, D.H. Friend, L.A. DaSilva, A.B. MacKenzie, Cognitive Networks, Springer, Netherlands, 2007.
[10] C. Fortuna, M. Mohorcic, Trends in the development of communication networks: cognitive networks, Comput. Netw. 53 (9) (2009) 1354–1376.
[11] D.H. Friend, Cognitive Networs: Foundations to Applications PhD thesis, Electrical and Computer Engineering Department, Virginia Tech University, Blacksburg, Virginia, 2009.
[12] R.W. Thomas, *Cognitive N*etworks PhD thesis, Electrical and Computer Engineering Department, Virginia Tech University, Blacksburg, Virginia, 2007.
[13] V. Srivastava, M. Motani, Cross-layer design and optimization in wireless networks, in: Q.H. Mahmoud (Ed.), Cognitive Networks: Towards Self-Aware Networks, John Wiley and Sons, 2007.
[14] A.S. Zahmati, S. Hussain, X. Fernando, A. Grami, Cognitive wireless sensor networks: emerging topics and recent challenges, in: Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference, IEEE, 2009, pp. 593–596.
[15] G. Vijay, E. Ben Ali Bdira, M. Ibnkahla, Cognition in wireless sensor networks: a perspective, Sensors J. IEEE 11 (3) (2011) 582–592.
[16] E. Meshkova, J. Riihijarvi, A. Achtzehn, P. Mahonen, Exploring simulated annealing and graphical models for optimization in cognitive wireless networks, in: Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE, IEEE, 2009, pp. 1–8.
[17] H.G. Goh, K. Hsiang Kwong, C. Shen, C. Michie, I. Andonovic, CogSeNet: a concept of cognitive wireless sensor network, in: Consumer communications and networking conference (CCNC), 2010 7th IEEE, IEEE, 2010, pp. 1–2.
[18] D. Singhal, S. Barjatiya, G. Ramamurthy, A novel network architecture for cognitive wireless sensor network, in: Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on, IEEE, 2011, pp. 76–80.
[19] M. Rovcanin, E. De Poorter, I. Moerman, P. Demeester, An LSPI based reinforcement learning approach to enable network cooperation in cognitive wireless sensor network, in: Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on, IEEE, 2013, pp. 82–89.
[20] A. Tena, G. Jara, J. Domingo, E. Romero, A. Araujo, Cognitive wireless sensor network platform for cooperative communications, Int. J. Distrib. Sens. Netw. 2014 (2014).
[21] M. Sujeethn, Quality of service routing algorithm for cognitive wireless sensor network, Bonfring Int. J. Netw. Technol. Appl. 2 (1) (2013) 01–09.
[22] J. Sen, Security and privacy challenges in cognitive wireless sensor networks, in: N. Meghanathan, Y.B. Reddy (Eds.), Cognitive Radio Technology Applications for Wireless and Mobile Ad Hoc Networks, IGI-Global, Hershey, Pa, USA, 2013, pp. 194–232.
[23] A. Fragkiadakis, V. Angelakis, E.Z. Tragos, Securing cognitive wireless sensor networks: a survey, Int. J. Distrib. Sens. Netw. 2014 (2014).
[24] M. Cardei, A. Mihnea, Channel assignment in cognitive wireless sensor networks, in: Computing, Networking and Communications (ICNC), 2014 International Conference on, IEEE, 2014, pp. 588–593.
[25] M. Thathachar, P. Shanti Sastry, Varieties of learning automata: an overview, Syst. Man. Cybern. Part B 32 (6) (2002) 711–722.
[26] K.S. Narendra, M.A.L. Thathachar, Learning Automata: An Introduction, Courier Corporation, 2012.
[27] S. Shenker, Fundamental design issues for the future Internet, Selected Areas Commun. IEEE J. 13 (7) (1995) 1176–1188.
[28] D. Raymer, S. van derMeer, J. Strassner, From autonomic computing to autonomic 5networking: an architectural perspective, in: Engineering of Autonomic and Autonomous Systems, 2008. EASE 2008. Fifth IEEE Workshop on, IEEE, 2008, pp. 174–183.
[29] K. Karenos, Traffic Management in Wireless Sensor Networks, University of California, Riverside, 2008.
[30] "Network Simulator" [online] http://www.isi.edu/nsnam/ns.
[31] M.A.L. Thathachar, P.S. Sastry, Networks of Learning Automata: Techniques for Online Stochastic Optimization, Springer Science & Business Media, 2011.
[32] M. Mirsadeghi, A. Mahani, Energy efficient fast predictor for WSN-based target tracking, Ann. Telecommun. 70 (1–2) (2015) 63–71.
[33] M.Z.A. Bhuiyan, G. Wang, A.V. Vasilakos, Local area prediction-based mobile target tracking in wireless sensor networks., Comput. IEEE Trans. 64 (7) (2015) 1968–1982.
[34] S. Vasuhi, V. Vaidehi, Target tracking using interactive multiple model for wireless sensor network, Inf. Fus. 27 (2016) 41–53.
[35] M. Li, Z. Li, A.V. Vasilakos, A survey on topology control in wireless sensor networks: taxonomy, comparative study, and open issues, Proc. IEEE 101 (12) (2013) 2538–2557.
[36] H. Safa, W. El-Hajj, H. Zoubian, A robust topology control solution for the sink placement problem in WSNs, J. Netw. Comput. Appl. 39 (2014) 70–82.
[37] S.M. Jameii, K. Faez, M. Dehghan, AMOF: adaptive multi-objective optimization framework for coverage and topology control in heterogeneous wireless sensor networks, Telecommun. Syst. (2015) 1–16.
[38] T. Amgoth, P.K. Jana, Energy-aware routing algorithm for wireless sensor networks, Comput. Elect. Eng. 41 (2015) 357–367.