

Designing Cognitive Wireless Sensor Networks Using Learning Automata

S. Gheisari

Department of Technology, Science and Research
Branch, Islamic Azad University, Tehran, Iran

S.gheisari@srbiau.ac.ir

M.R. Meybodi

Computer Engineering and Information Technology
Department, Amirkabir University of Technology,
Tehran, Iran.

mmeybodi@aut.ac.ir

Abstract- Adding cognition to the existing Wireless Sensor Networks (WSNs) with a cognitive networking approach brings about many benefits. 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. To the best of our knowledge, almost all of 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 we discuss the design of optimization algorithms for CWSNs using learning automata. Proposed algorithms select appropriate protocols and carrying out learning automata on the resulting stack in order to maximize the perceived network performance. We test our algorithms using the ns-2.35 simulation platform. Our results show that learning automata approach works well to apply cognition in WSNs.

Keywords- Cognitive Wireless Sensor Network; Learning Automata

I. 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 are capable of sensing, communicating, and can be developed at a cost much lower than traditional wired sensor systems. Designing WSNs are facing many challenges; energy efficiency and life time maximization are two key design factors because the battery capacity is limited and is not chargeable in WSNs. Spectrum allocation is another challenge 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. Collusion occurs in WSNs because of the hidden terminal problem, whereupon control and data packet may lose so managing the 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], [3]. Proposed solution which uses cognitive radio is fundamentally new approach to the spectrum allocation and utilization concept. As it is obvious, a cognitive radio applies cognition only at the physical layer to dynamically detect and uses spectrum holes, focusing strictly on dynamic spectrum access, whereas the objective of cognitive networking [4], [5] is to apply cognition to the entire network protocol stack for achieving network-wide performance goals. In a Cognitive Wireless Sensor Network (CWSN), sensor nodes would have

additional state called *sensing state* [6] where they sense the environment and use their observation to tune their controllable parameters from different protocols of protocol stack in order to adapt with environment and satisfy users' goals. It has shown that the coexistence of such networks can significantly degrade a WSN's performance. Maximizing the received network performance by selecting appropriate parameters and carryout cross-layer optimization on the resulting stack is a key functionality of any CWSN. Moreover by the adaptive changes of system parameters different data rates can be achieved which in turn can directly influence the power consumption and the network lifetime.

In this study, we focus on the optimization of the composition and configuration of the entire protocol stack likewise [7], [8] but by developing learning automata based algorithms and limiting our work to WSNs. In proposed algorithms, a learning automaton is assigned to each controllable parameter in a protocol stack to tune the parameters in order to increase the network-wide performance. We also specify how the learning task can be stated using a utility function as a common reinforcement signal for all learning automata, or using different feedbacks from the environment as reinforcement signals for each automaton in a contest manner.

The rest of the paper is structured as follows. In section two we formulate WSN protocol stack composition and formulation as a learning problem and describe the overall architectures of our proposed cognitive algorithms, also Learning Automata is explained briefly. Cognitive algorithms and our WSN scenario are discussed in section three. In section four performance evaluation and the experimental results are presented. Finally we conclude the paper in section five.

II. Problem Formulation¹

Network protocols have controllable and observable parameters. Observable parameters consisting application level parameters as end-to-end goals are affected by controllable parameters. Examples of controllable parameters in a CWSN are the maximum number of transmission in MAC protocols as well as sensing interval, beaconing interval or duty cycle. Common examples of observable parameters are the number of retransmission in MAC protocols as well as network delay, throughput, power consumption or reliability. Such representations allows for a simple control loop in which controllable parameters are tuned in accordance with observable parameters to improve the performance of whole network. The quality of observable parameters can be measured individually or using an application specific utility function [9]. 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 the flexible way to define satisfaction; they allow for a quantifiable expression of user needs [10] and can be used by controllable parameters to tune themselves. However utility functions are known to be difficult, they embody most of the objective functions in networking used today such as QoS classes. 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.

¹ In this section we widely use reference [7] formulations

In this study the controllable parameters adjustments are conducted by learning automata. The overall system models are represented in Figure 1 and Figure 2.

A. Assumptions

WSNs and also CWSNs can be characterized by their applications or objectives that their users want to achieve. For simplicity we take a centralized view on, i.e., all nodes use the same parameter values and all nodes evaluate their parameter in the same way. More over network parameters are likely to be optimized at different time scales, for example switching between sensor states is likely to take longer than adjustment of a single MAC parameter; but we do not consider this fact. Such considerations will lead to even more effective learning.

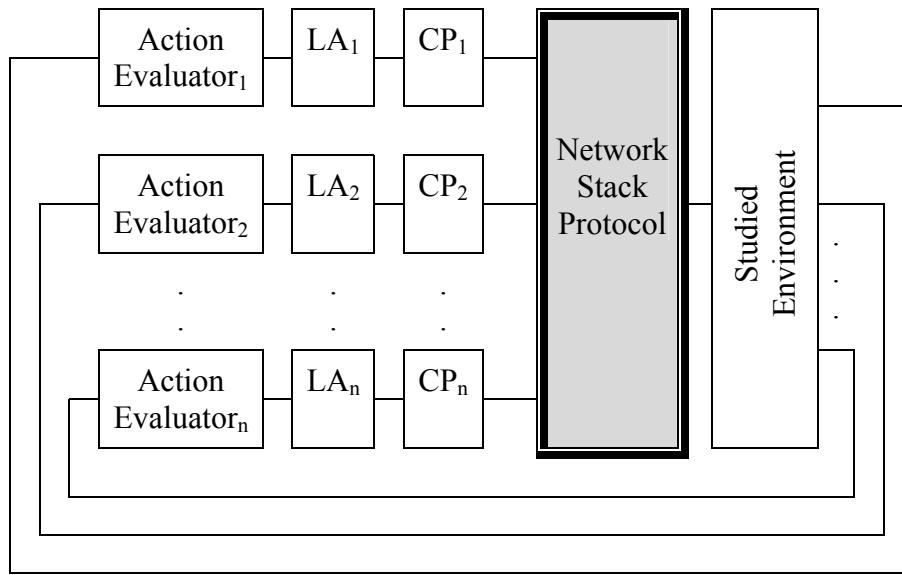


Fig. 1 architecture of CWSN without using a unique utility function

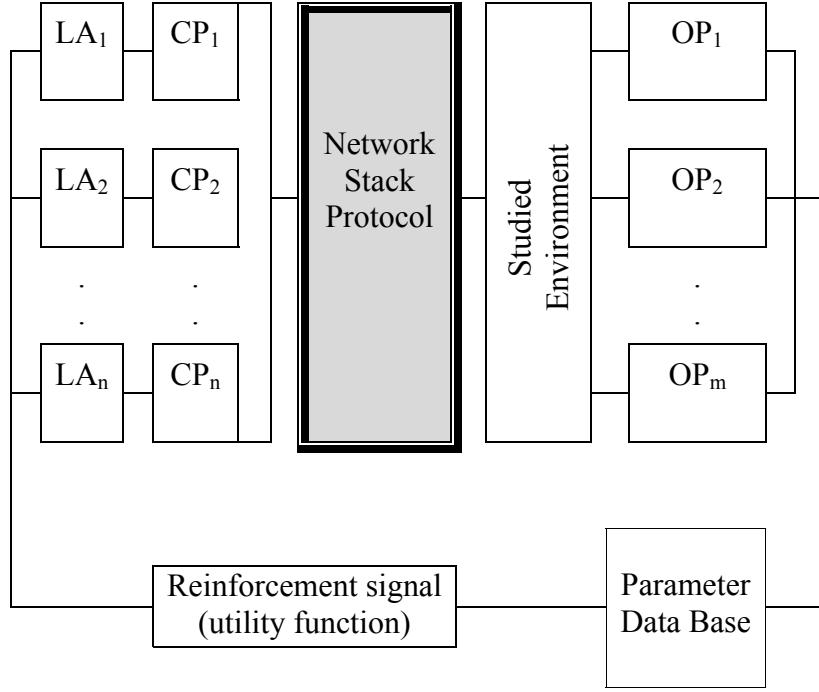


Fig. 2 architecture of CWSN with using a unique utility function

B. Formal description

We can consider CWSN designing problem as a protocol stack learning problem, which is a search among all permutation of parameter values of network protocol to maximize the network performance. The formal description of the problem is as follows. Consider S as the collection of all possible configuration of CWSN protocol stacks. Each stack 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 user constraints. The set of stacks satisfying the constraints is simply defined by $S_c = \{x \in S \mid c(x) = 1\}$

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

The performance of the stack and its characteristics is described by a set of observable parameters OP . this set is not necessarily defined only from a technical point of view. Economic considerations such as cost of deploying a certain topology, bandwidth-based fees and other application-specific parameters may also be taken into account. We study observable parameters in two manners. First (see Figure 1) we classify them corresponding with each controllable parameter, and then evaluate their values individually to feed the controllable parameters and desirable protocol stack configuration can be described as the

result of separated evaluations. Second (see Figure 2) we evaluate the network performance based on all observable parameters values. The set of observable parameter values need to be weighed against each other. This is usually achieved by applying an utility function U ($OP(s, cp, E)$), that maps the parameter set to a single numerical, taken into consideration the deployed stack ($s \in S$, $cp \in CP$) and the operational environment E . Higher values of utility determine a performance of the users of one set of controllable parameter values over the other. In this study we use reinforcement signal in learning automata instead of utility function and desirable configuration can be described by equation (1).

$$(s, cp) = \operatorname{argmax}_{(s, cp) \in (S_c \times CP_{c_{S_c}})} \beta(OP(s, cp, E)) \quad (1)$$

The maximization task is to find the stack with a valid combination of parameters that yields the highest overall reward.

C. Learning Automata

Learning automaton is an adaptive decision making device that tries to learn the optimal action from a set of allowable actions in an unknown random environment [11]. This process is done by interacting between an automaton and a random environment. At each time t , the automaton selects an action, $\alpha(t)$ using its action probability distribution, $p(t)$, and applies the selected action to the random environment. Then, the random environment generates a stochastic reinforcement signal, $\beta(t)$, to the automaton [12]. The automaton updates the action probability distribution using both the reinforcement signal and a learning algorithm. Figure 3 represents the interaction between a learning automaton and its environment. Learning automaton based on its action-set can be classified into two major classes: finite-action-set LA (FALA) and continues-action-set LA (CALA) which has actions chosen from real line. For an r -action FALA, the action probability distribution is represented by r -dimensional probability vector and is updated by the learning algorithm. Learning automata have been used successfully in many applications such as routing and call admission control in computer network, solving NP-Complete problems, and too many other applications. For further information about LA please refer to [12].

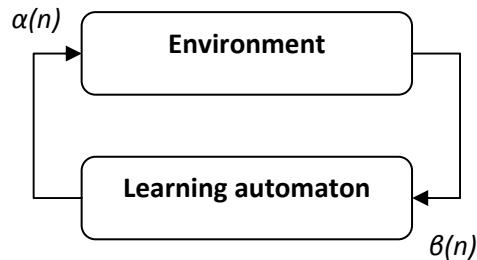


Fig. 3 the interaction between learning automaton and the environment

A single automaton is generally sufficient for learning the optimal value of one parameter. However for multidimensional optimization problems, we need a system consisting of as many automata as there are parameters [11]. Let A_1, \dots, A_N be the automata involved in an N -player game. Each play of game consists of each of the automata players choosing an action and then getting the *payoffs* or reinforcements from the environment for this choice of actions by the group of LA. Let $p_1(k), \dots, p_N(k)$ be the action probability distributions of N automata. Then at

each instant k , each of the automata A_i chooses an action $\alpha^i(k)$ independently and at random according to $p_i(k)$, $1 \leq i \leq N$. This set of N actions is input to the environment which responds with N random payoffs, which are supplied as reinforcements to the corresponding automata. Special case of this model is a game with *common payoff*. Here all the automata get the same payoff from the environment (that is, $\beta^i = \beta$). Hence there is only one payoff function, e.g., d . LA in a common payoff game is often referred as a team of LA.

D. Properties of CWSNs designing

Designing CWSNs using learning automata has several properties that distinguish it from classical WSNs optimization. First, it does not need any definite starting point - the default setting of the controllable parameters of the protocols that form the stack. In the beginning all the controllable parameters values have the same chance to be chosen and in each iteration their probabilities of be chosen update according to the received reinforcement signal(s). This prevent of sticking local optima. However it may cause execution time overhead.

The second peculiarity of adding cognition to WSN is a distinction of two phases: offline and online. The offline stage can be regarded as a long-term learning, i.e. the network can be observed for a long time without the need to keep high average reward during the training period. This stage can take place at times of low network load, for example, during night times or at the stage of pre-deployment network testing. It allows for the best network optimization; however this period is not available to all systems. The online stage has more constraints. First it must have shorter learning periods not to disturb the user. Second, reinforcement values must be kept high for user satisfaction. For the same reason the system must react fast to change network conditions, which can be stated as inferring in cognitive networks. If the reinforcement signal increases then appropriate parameter value sets should be detected as well. Ideally in a cognitive network we have both of these phases: first conducting the offline training as a preliminary learning, for example limiting the number of possible parameter set permutations, and then continuously execute online learning as a background process on the working system to make instance decisions.

Finally, usually there exists a strong dependency of controllable parameters with observable parameters for a wide range of network scenarios (including different topologies and application traffic patterns). The exception is drastic change in the network's performance. When contribution of controllable parameters via observable parameters to the final utility changes. As an example there might be a sudden increase in the node failure rate when the utility focus on shifts from low-power message delivery to just message delivery. In this exception CWSN should recognize the change and react immediately.

III. Cognitive Algorithm

In this section we describe learning automata based algorithms in order to add cognition to WSNs. We define a maximum iteration number as an additional limitation to the learning. If the algorithm is unable to find better results within a certain number of steps, it stops the search and returns to the parameter values that yielded highest utility.

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

Since our training is to be applied at runtime, we have been further trying to limit the negative impact of the online learning to the users. As the algorithms (especially during early stages) try for parameter combinations that yield comparably low utility, we add several conditional barriers to the algorithms. This way we were able to cope with high fluctuations in utility. For example, once the LA converges to a good stable solution, the training is stopped. The observable parameters or the utility are continuously monitored over time. If the system notices a sudden drop in performance that cannot be traced back to short-time variances but indicates a change in network usage, LA 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 training period and the average utility achieved during this time. The training iterations may be conducted only every N^{th} iteration, for all other returning to the best state. This allows limiting the negative experience for the user at expense of a prolonged convergence time.

A. Scenario and Implementation

WSNs and CWSNs have low-power radios as their performance characteristics because as mentioned before, their nodes battery capacity is limited and is not chargeable. The performance of WSNs is known to be extremely sensitive to the deployment environment, and therefore, it is feasible to conduct extensive static, pre-deployment tests of these networks. This is due to the fact that signals of low-power radios are easily disrupted. Once the network has been deployed at the location, the task of automatic node configuration with the purpose of its fine-tuning to the surrounding environment is therefore highly relevant. This is known as topology control in WSNs and we will study in further researches.

Additionally, there exist no widely accepted well-performing protocol stacks for these networks, and multiple protocols and multiple parameters might be considered for deployment. Here we consider six controllable parameters. In application layer sensing interval determines the durations of sleep and awake for sensor nodes. Beaconing interval and beaconing time out, in routing layer, define respectively the time interval and lifetime for signaling packets. In MAC layer we consider duty cycle to define the duration of send and receive; contention window to determine the domain of waiting interval before retransmissions; and max-retransmit which limits the retransmission attempts after failure. Each parameters values can be chosen from a finite set of possible values which are obtained from previous network monitoring in the case of using finite action set learning automata; or can be sampled from valid intervals in the case on continues action set learning automata.

Here, a learning automaton with finite action set (FALA) is assigned to each controllable parameter. Each automaton selects a value from its action set corresponding with its parameter; the list of selected actions forms the protocol stack. Then environment feedback which is obtained from observable parameters is used to update the probability vectors of learning automata. The algorithms update the action probability distribution of all automata using following equation:

$$p^j(n+1) = p^j(n) + \lambda \beta(n) \left(e_{\alpha_j} - p^j(n) \right) \quad j = 1, \dots, m \text{ (for all actions in action set)}$$

Where $0 < \lambda < 1$ is a constant parameter which is called learning rate; e_{α_j} is a unit vector of appropriate dimension, corresponding with the number of action in action set, with α_j th component unity. And finally $\beta(t) \in [0,1]$ is reinforcement signal or environment feedback to learning automata and we use normalized value of observable parameters or utility to define it. This algorithm is known as Linear Reward-Inaction (L_{R-I}) algorithm. All above steps should be repeated until for max_{it} iterations observable parameters or utility don not improve.

As mentioned in problem formulation we suggest two architectures for CWSN, one of them uses observable parameters directly as environment feedback which is called DF-LA (see Figure 1), whereas other uses a function to compute utility from observable parameters which is named U-LA (see Figure 2).

Observable parameters are power consumption, packet-loss, throughput, end-to-end delay, and number of retransmissions, before a successful transmit. So utility is a function of power, reliability, throughput, and average delay.

IV. Performance Evaluation

Both cognitive algorithms for CWSN are implemented in NS-2.35 [13]. We have evaluated the performance of designed CWSN over a wide range of scenarios. Network sizes varied from two hundred to five hundred nodes with random topologies. Network is queried every N milliseconds and get information on node performance, and parameters of the nodes are adjusted as result of decision and learning processes.

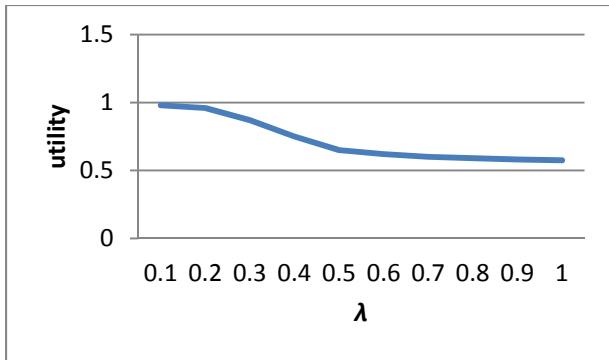
We classify our experiments in to two categories: sensitive experiments, comparative experiments. In the first category we study the effect of each parameter on the performance of designed CWSN. Comparison the performance of proposed architecture with each other and with WSN without cognition is done in the second category of experiments.

A. Sensitive analysis

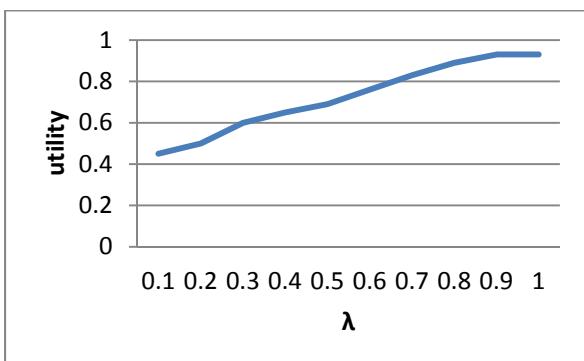
In this section we study the effect of each parameter on the performance of designed CWSN. The proposed algorithms have two parameters: (1) constant parameter in the learning rule of learning automata (λ), (2) the number of iterations, after it we stop learning if the performance does not improve (max_{it}). In order to study the effect of one parameter, the value of other parameter is fixed. The default values of parameter are: $\lambda=0.3$ and $max_{it}=40$. In what follows, we study the effect of each parameter on the performance of the algorithms.

The effect of parameter λ on the performance of algorithms is analyzed for three learning automata corresponding with controllable parameters in application and network layers, and other learning automata. Also, it is analyzed for DF-LA and for U-LA separately. Utility is used as performance scale. The results are represented in Figure 4.

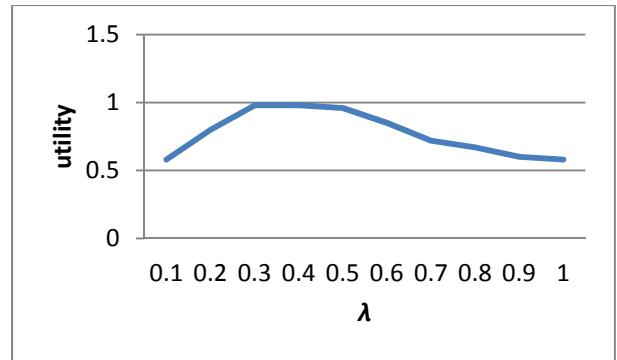
Finally, the effect of max_{it} on the performance of algorithms is analyzed in the same manner with parameter λ experiments. Figure 5 shows the results.



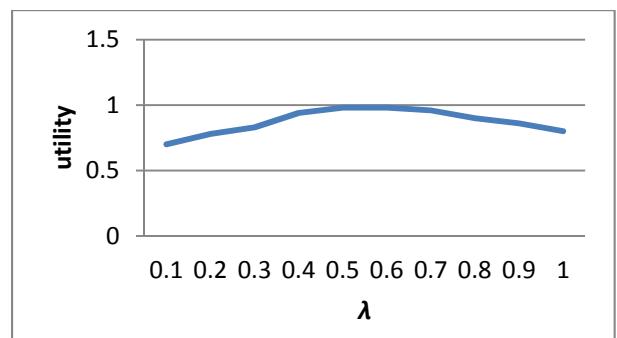
(a) Effect of the parameter λ on DF-LA for learning automata corresponding with MAC layer parameters



(b) Effect of the parameter λ on DF-LA for learning automata corresponding with application and network layers parameters

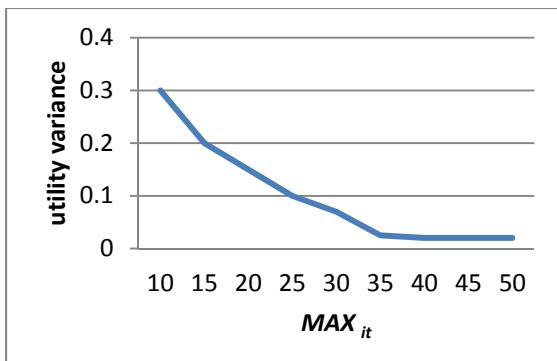


(a) Effect of the parameter λ on U-LA for learning automata corresponding with MAC layer parameters

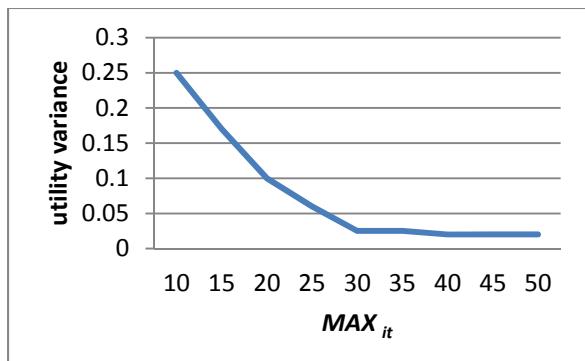


(b) Effect of the parameter λ on U-LA for learning automata corresponding with application and network layers parameters

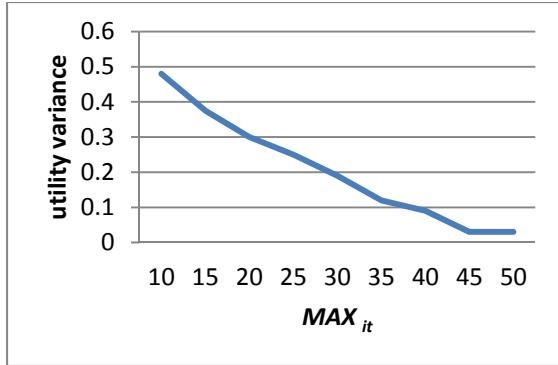
Fig. 4 The effect of parameter λ on the performance of algorithms



(a) The effect of max_{it} on the performance of DF-LA for learning automata corresponding with MAC layer parameters
(b)



(c) The effect of max_{it} on the performance of U-LA for all learning automata



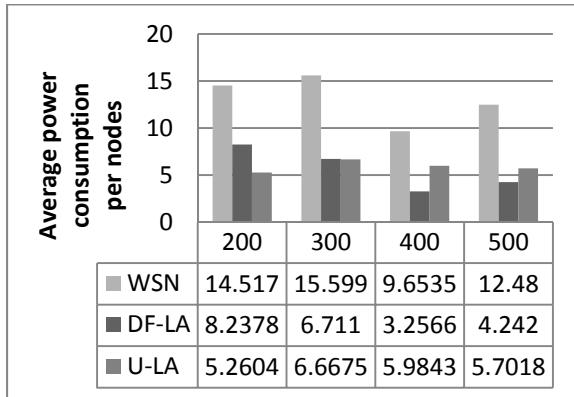
(b) The effect of \max_{it} on the performance of DF- LA for learning automata corresponding with application and network layers parameters

Fig. 5 the effect of \max_{it} on the performance of algorithms

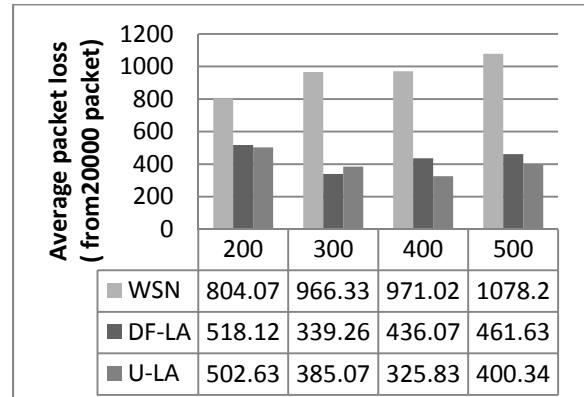
B. Comparison the DF-LA and U-LA with WSN without cognition

In this section the performance of proposed algorithms are compared with each other and with the performance of WSN without cognition. Figure 6 represents the results. Power consumption, delay, packet loss, throughput, received packet, and finally utility as the function of them are observed.

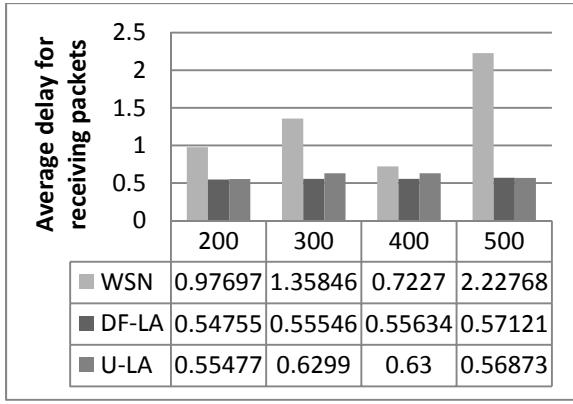
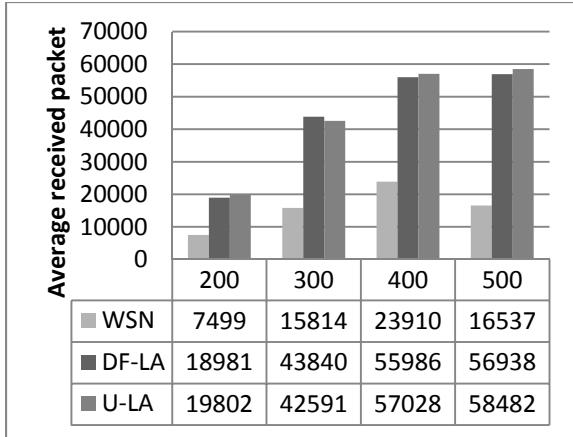
As it can be seen in the charts, CWSN algorithms always have better performance in comparison with WSN without cognition. It was predictable since CWSN tune the parameters in order to improve the performance whereas WSN without cognition adjust the parameters with no attention to the environment conditions.



(a) Power consumption

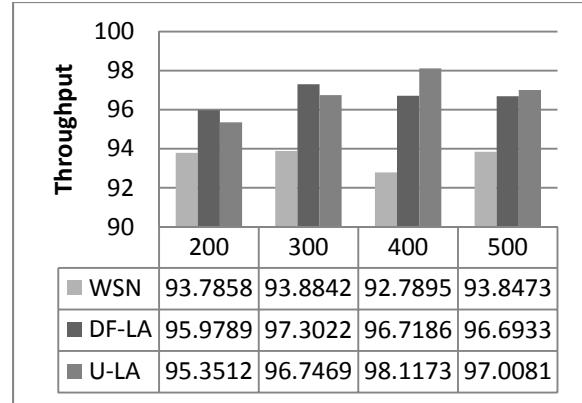


(c) Packet loss

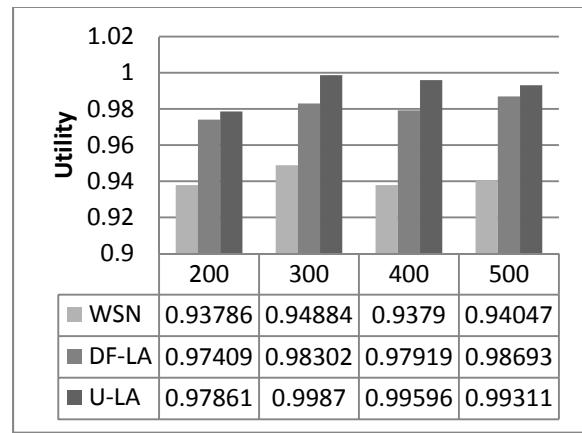


(b) Delay

(e) Delivered packets



(d) Throughput



(f) Utility

Fig. 6 Comparison the performance of cognitive approaches in WSN with basic WSN (the charts are obtained by conducting multiple experiments at random starting parameter configurations and then averaging over these results is done.)

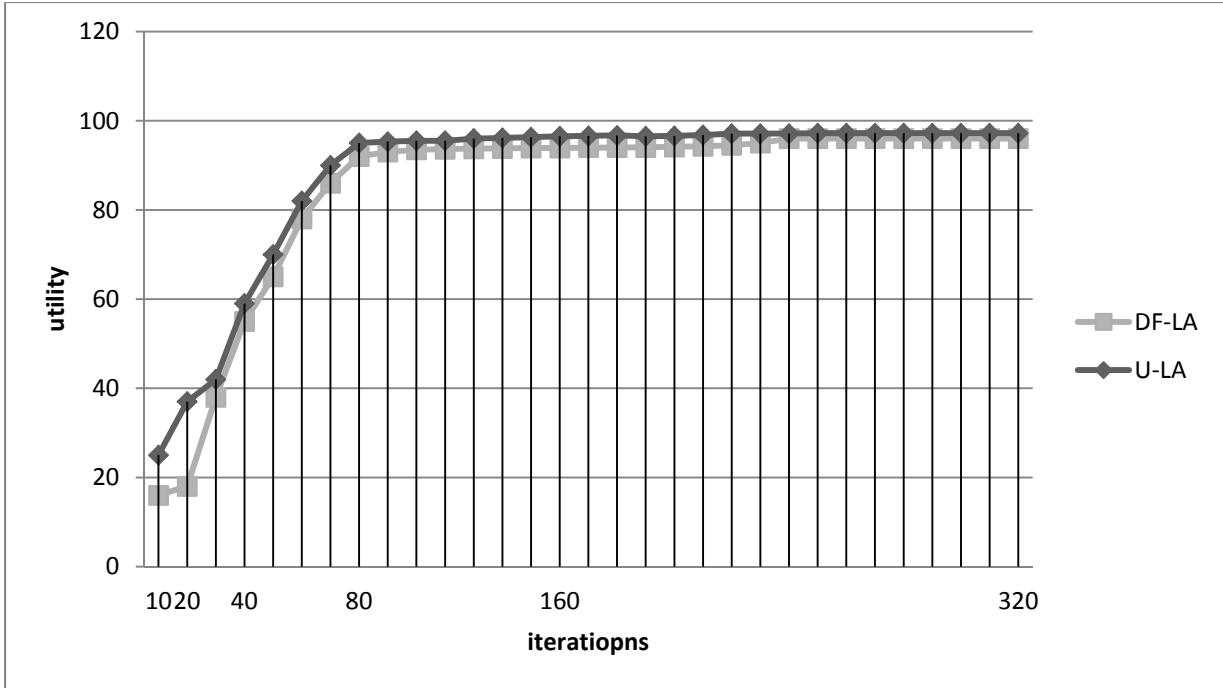


Fig. 7 The convergence speed

C. The speed of convergence of CWSN algorithms

The result of evaluation and comparison of the convergence speed is presented in Figure 7. As it can be seen U-LA converges to 95% from maximum after approximately 70-80 iterations; and it takes 80% of maximum within 50-60 iterations. DF-LA converges to 95% from maximum after 160 iterations but it takes about 90% within 70-80 iterations.

V. Conclusions

In this study we have considered adding cognition to WSN and discussed its properties. We proposed two algorithms, DF-LA and U-LA which used learning automata to tune the networks controllable parameters. Each learning automaton was assigned to one parameter and chose one possible value for it. In the evaluation phase DF-LA evaluated automata separately and with different scales; whereas U-LA used the unique utility function to produce the reinforcement signal for all automata. In other words it can be said that DF-LA is an asynchronous scheme and U-LA is a synchronous scheme. Proposed CWSNs has successfully tested. The results represented the acceptable performance of them. However runtime learning may offer users and CWSN catches the acceptable performance after some iteration, in the network with long lifetime its high performance is worthy.

In this study we did not consider the parameter of congestion control in transport layer; considering congestion control parameter which can form a complete view of traffic control problem in WSNs will be our future plan of research.

References

- [1] Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal. "Wireless sensor network survey." *Computer networks* 52, no. 12 (2008): 2292-2330.
- [2] Cavalcanti, Dave, Sushanta Das, Jianfeng Wang, and Kiran Challapali. "Cognitive radio based wireless sensor networks." In *Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on*, pp. 1-6. IEEE, 2008.
- [3] Gao, Song, Lijun Qian, and Dhadesugor R. Vaman. "Distributed energy efficient spectrum access in wireless cognitive radio sensor networks." In *Wireless communications and networking conference, 2008. WCNC 2008. IEEE*, pp. 1442-1447. IEEE, 2008.
- [4] Thomas, Ryan W., Daniel H. Friend, Luiz A. DaSilva, and Allen B. MacKenzie. *Cognitive networks*. Springer Netherlands, 2007.
- [5] Fortuna, Carolina, and Mihael Mohorcic. "Trends in the development of communication networks: Cognitive networks." *Computer Networks* 53, no. 9 (2009): 1354-1376.
- [6] Zahmati, Amir Sepasi, Sattar Hussain, Xavier Fernando, and Ali Grami. "Cognitive wireless sensor networks: emerging topics and recent challenges." In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pp. 593-596. IEEE, 2009.
- [7] Meshkova, Elena, Janne Riihijarvi, Andreas Achtzehn, and Petri Mahonen. "Exploring simulated annealing and graphical models for optimization in cognitive wireless networks." In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pp. 1-8. IEEE, 2009.
- [8] Quer, Giorgio, Hemanth Meenakshisundaram, Bheemarjuna Tamma, B. S. Manoj, Ramesh Rao, and Michele Zorzi. "Cognitive Network Inference through Bayesian Network Analysis." In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1-6. IEEE, 2010.
- [9] Shenker, Scott. "Fundamental design issues for the future Internet." *Selected Areas in Communications, IEEE Journal on* 13, no. 7 (1995): 1176-1188.
- [10] Raymer, David, Sven van der Meer, and John Strassner. "From autonomic computing to autonomic networking: an architectural perspective." In *Engineering of Autonomic and Autonomous Systems, 2008. EASE 2008. Fifth IEEE Workshop on*, pp. 174-183. IEEE, 2008.
- [11] Thathachar, M., and P. Shanti Sastry. "Varieties of learning automata: an overview." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 32, no. 6 (2002): 711-722.
- [12] Narendra, Kumpati S., and Mandayam AL Thathachar. *Learning automata: an introduction*. Courier Corporation, 2012.
- [13] "Network Simulator" [online] <http://www.isi.edu/nsnam/ns>