Computer Networks 124 (2017) 11–26

Contents lists available at ScienceDirect

Computer Networks

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

A new reasoning and learning model for Cognitive Wireless Sensor Networks based on Bayesian networks and learning automata cooperation

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

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

ARTICLE INFO

Article history:
Received 4 May 2016
Revised 26 March 2017
Accepted 30 May 2017
Available online 1 June 2017

Keywords:
Bayesian Networks
Cognitive networks
Learning automata
Reasoning
Wireless Sensor Network

ABSTRACT

Adding cognition to existing Wireless Sensor Networks (WSNs) with a cognitive networking approach, which deals with using cognition to the entire network protocol stack to achieve end-to-end goals, brings about many benefits. However cognitive networking may be confused with cognitive radio or cross-layer design, it is a different concept; cognitive radios applies cognition only at the physical layer to overcome the problem of spectrum scarcity, and cross layer design usually focuses on linking at least two non-consecutive specific layers, to achieve a particular goal. Indeed, it can be said that the cognitive radio and the cross layer design are two effective methods in cognitive networking. 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 in the physical layer. In this paper, we propose a new reasoning and learning model for CWSNs, in which firstly, a team of learning automata is employed to construct a Bayesian Network (BN) model of the parameters of the network protocol stack, and then the constructed BN is used to tune the controllable parameters. The BN represents the dependency relationships between the parameters of the network protocol stack, and the BN-based reasoning is an efficient tool for cross-layer optimization, in order to maximize the perceived network performance. Simulations have been done to evaluate the performance of the proposed model. The results of the simulations show that the proposed model successively adds cognition to a WSN and improves the performance of the communication network.

1. Introduction

Wireless Sensor Networks (WSNs) consist of hundreds or even thousands of small devices each with sensing, processing, and communication capabilities to monitor a real-world environment. They are envisioned to play an important role in a wide variety of areas, arranging from critical military surveillance applications to forest fire monitoring [1]. Designing a WSN faces many challenges; here, some of them are mentioned. Usually, the power resource of a WSN is batteries, which are limited and un-chargeable. As a result of power limitation, sensor nodes have low-power radios, which mostly use unlicensed ISM bands (2.4 GHz.), or Wi-Fi, Bluetooth, etc. with the same band. In addition, the performance of WSNs is extremely sensitive to the deployment environment, because the signals of the low-power radios are easily disrupted. Finally, there exist no widely accepted well-performing protocol stacks for these networks, and multiple protocols and multiple parameters might be considered for deployment.

Cognitive radio has emerged as a key technology to solve the problem of spectrum utilization. A cognitive radio is an intelligent wireless communication system that is aware of its surrounding environment, and adapts its internal parameters to achieve a reliable and efficient communication and optimum utilization of the resources [2]. With the advent of cognitive radio technology and cross-layer design [3], Thomas et al. [4] have introduced a different perspective of the traditional networks, named *cognitive network*. Cognitive network is a communication network with a cognitive process that can perceive current network conditions, then it can plan, decide, and act on those conditions, and also learn from the consequences of its actions; all while following end-to-end goals [5]. Using the concept of cognitive, Cognitive Wireless Sensor Networks (CWSNs) have been presented, in which sensor nodes can change their configurations according to the environment conditions. Cognitive capabilities are based on following activities: mon-

* Corresponding author.
E-mail addresses: so_gheisari@pardisiau.ac.ir, so_gheisari@yahoo.com (S. Gheisari), mmeybodi@aut.ac.ir (M.R. Meybodi).

http://dx.doi.org/10.1016/j.comnet.2017.05.031
1389-1286/

itoring of the environment, analysis and characterization of the environment, optimization of the best communication strategy based on different constraints such as reliability, power, etc., and adaptation and collaboration strategy. The CWSNs can overcome the mentioned designing challenges in the WSNs and it can also provide more reliable communication channels while reducing power consumption, increasing network life-time and reliability, and also enhancing *quality of service* (QoS) guarantee to applications.

A key feature of the cognitive networks, as described in [4], is the cognitive process, which is responsible for the reasoning and learning. The terms reasoning and learning are very close together. We consider reasoning as the immediate decision process that must be execute using available historical knowledge as well as knowledge about the current state of the world; but, learning is a long-term process consisting of the accumulation of knowledge based on the perceived results of past actions. Generally, cognitive nodes learn by enriching a knowledge-base to improve the efficacy of future reasoning [6]. Many methods may be used for learning and reasoning in a cognitive network. Some of the generic methods for reasoning are: *Distributed Constraint Reasoning, Bayesian Networks*, and *Meta-heuristics* such as *Genetic Algorithm*; and some of the methods for learning are: *Q-learning, Learning Automata*, and *Bayesian Networks* [4].

A *Bayesian Network* (BN) is a method of reasoning under uncertainty, which may be a result of limited observations, noisy observations, unobservable states, or uncertain relationships between inputs, states, and outputs within a system [7]. All of these causes for uncertainty are common in communication networks. The ability of cognitive networks to potentially control the parameters at different layers in the protocol stack gives rise to concern over interactions between different protocol layers; interactions that are currently not well understood [8]. Therefore, understand the dependencies between these protocols, and specially their parameters, is very important and also sophisticated; and we need a tool that can encode these dependencies. Approximate probabilistic graphical models, like BN, provide a means for dealing with this uncertainty and dependency. BNs decompose a joint probability distribution (JPD) over a set of variables (i.e. events) into a set of conditional probability distributions defined on a directed acyclic graph (DAG). Each node of the DAG represents a variable in the JPD. The directionality of the edges in the DAG represents parent-child relationships, where conditioned on the knowledge of its parents, a child is independent of all other non-descendants in the network [7]. Each node contains a conditional probability distribution, which is the probability distribution of the variable at that node conditioned on the variables at all parents of the node. The JPD can then be reconstructed as the product of all of the conditional probability distributions.

Another advantage of the BN for cognitive networks is that, BN is an example of a method that incorporates both reasoning and learnings, simultaneously. Learning in a BN is accomplished through belief updating. Belief updating is the process of modifying the conditional probability distributions based on observations of variable in the JPD. Knowledge is contained within the conditional probability distributions as well as the DAG structure itself. Reasoning in a BN consists of producing probability estimates of sets of variables based on the current JPD and possibly observed variables as well. However, BNs do not specify a method for selecting an action and they may be paired with a decision making method such as multi-criteria decision making [9].

Before using the BNs as the reasoning and learning tool in a cognitive network, some problems must be solved; for example, the structure of the dependency graph may be unknown a priori. An obvious consequence of this is that the conditional distributions are also unknown. These issues are normally addressed by learning the structure of the DAG and then estimating the conditional distributions. These processes require a set of data consisting of past combinations of inputs and outputs of the communication network. This data consists of samples of the network's behavior, and there is no concept of a correct output for any particular input. Learning a BN from data imposes a heavy computational burden, and this has led researchers to find ways to parallelize the process. For structures of moderate dimensionality, an asynchronous complete search such as in [11] may be feasible. Researchers have also used meta-heuristics such as ant colony optimization [12], variable neighborhood search [13], and evolutionary algorithms [14] to provide approximate solutions to the BN structure.

However, despite this complexity, the ability to learn the structure and conditional distributions of a BN allows a cognitive network to construct a set of beliefs about the network conditions by observing the previous behaviors of the network and feedbacks of its environment. This is very beneficial for networks that may be deployed in a variety of scenarios, but cannot be fully characterized beforehand; one example of these communication networks is wireless sensor network or WSN. An additional benefit of learning a BN is that prior knowledge (such as that generated by an expert) can be incorporated into the BN to improve the learning process [10].

In this study, we propose a BN based model for reasoning and learning in cognitive networks. After constructing, the BN represents the dependence relationship among the parameters of the network stack protocols and it can perform probabilistic inference in the communication network. Nevertheless as mentioned before, structure learning of BNs is a problem belongs to the class of NP-hard problems and typically greedy algorithms are used to solve it; so in the proposed model, firstly, a team of learning automata is applied to construct the BN, using training samples [15–16]. Using the learning automata leads to find an optimal (or near-optimal) network in lower computation time. Then, after a simple parameter learning, a BN inference is used for optimizing the composition and the configuration of the entire protocol stack in a cross-layer view of the communication network.

The rest of the paper is structured as follows. In Section 2 preliminaries, such as CWSN features, BN, and learning automata, are briefly explained. In Section 3 proposed reasoning and learning model is discussed, and also a WSN scenario is introduced and reformed to the CWSN. The performance evaluation and the experimental results are presented in Section 4. Finally, we conclude the paper in Section 5.

2. Preliminaries

In this section, firstly, we will discuss the features of CWSNs. Then, BNs will be described as a graphical model for representing conditional independences between the variables through a DAG. Also structure learning, parameter learning and inference in BNs will be explained. Finally, we will summarize learning automata as an intelligent tool for BNs structure learning.

2.1. Features of CWSNs

Cognitive networks, in general, and the CWSNs, in particular, consist of controllable and observable parameters. Controllable parameters are tunable parameters that their configurations affect the performance of the network. On the other hand, observable parameters are the parameters, which state the network performance. Examples for controllable parameters are: the duration of sensing interval in the application level, the value of contention 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), and the number of retransmitted packets. Such representations allows for a simple control loop, in which controllable parameters are tuned in accordance with observable parameters to improve the performance of the whole network. So, a model, which can represent the relationship between observable and controllable parameters, is essential. We propose a model based on BNs, which is also used as the reasoning and learning tool.

2.2. Bayesian networks

As mentioned before, BNs are widely used for knowledge representation and reasoning under uncertainty in intelligent systems. A BN describes the joint probability distribution (JPD) over a set of random variables, with defining a series of probability independences and a series of conditional independences [9]. From an informal perspective, BNs are directed acyclic graphs (DAGs), where the nodes are random variables, and the arcs specify the independence assumptions that must be held between the random variables. The joint probability of any particular combination of n random variables can be written according to Eq. (1) [15].

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(x_i | Parents\ (X_i)), \tag{1}$$

where $(X_1, \ldots, X_n)$ is the set of random variables. When the network is constructed, it can be an efficient device to perform probabilistic inference; however, the problem of constructing such a network remains. Giving a dataset of training samples, construction the BN can be separated into two subtasks: structure learning to determine the topology of the network, and parameter learning, which defines the numerical parameters (conditional probabilities) for a given network topology [17].

2.2.1. Structure learning in BNs

BN structure learning has two main approaches: constraint based approach [18–19], and search and score based approach [11–17] and [20–22]. In this work, we focus on structure learning based on search-and-score approach. Algorithms, which are following this approach, have two main components: a search procedure and a scoring metric. Search procedure determines an algorithm to search throughout all possible networks and select one; while scoring metric evaluates the quality of the selected network.

Scoring metrics: There exist a variety of metrics, which are proposed for evaluating a BN constructed network, such as Bayesian Metrics, Minimum Description Length (MDL), and Bayesian Information Criterion (BIC) to mention a few. Bayesian Metrics measure the quality of BNs by computing a marginal likelihood of the BNs, with respect to the given data and inherent uncertainties [23–24]. MDL is another type of scoring metric based on information theory and data compression [25]. The Bayesian information criterion (BIC), which will be used in this study, is a criterion for model selection among a finite set of models, and it is based on penalizing maximum likelihood. The BIC scoring metric for the constructed graph G is given by the following equation:

$$BIC = \log_2 P\left(D|G,\ \hat{\theta}_G\right) - \frac{|n|}{2}\log_2(M), \tag{2}$$

where D is the set of training samples; $\hat{\theta}_G$ is maximum likelihood (ML) estimation for G's parameters, and M is the number of training samples in the dataset. If all variables are multinomial, by considering r_i as a finite set of outputs for x_i, q_i as the number of configurations for x_i's parents, N_{ij} as the number of observations for x_i if j is its parents configuration, and finally N_{ijk} as the number of observation for $x_i = k$ when its parents configuration is j, Eq. (2) can be rewritten as following,

$$BIC = \sum_{i=1}^{M}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} log_2\left(\frac{N_{ijk}}{N_{ij}}\right) - \frac{log_2 N}{2}\sum_{i=1}^{M} q_i(r_i - 1). \tag{3}$$

In this case BIC converts to a simple counting problem [15–16].

Search methods: Given a scoring metric, the problem of learning the structure of a BN belongs to the case of NP-hard problems and there is no polynomial-time algorithm for finding the best network structure corresponding to the most scoring metrics [26]. Usually, a simple greedy algorithm is used to build the network. The greedy algorithm can add, reverse, or delete an edge, with the greatest improvement of the current network quality in search step, until no more improvement is possible. The initial network structure can be a graph with no edges, or it can take advantage of using prior information, such as using the best tree computed by the polynomial-time maximum branching algorithm [27–28]. BNs are acyclic graph, so after each iteration, the constructed network is validated and all cycles are removed.

2.2.2. Parameter training in BNs

Parameter training is another subtask of BNs learning, which defines the numerical parameters (conditional probabilities) for a given network topology. According to the definition of BNs, each variable is directly conditioned only by its parents, so the estimation of the parameters for each variable should be performed conditioned on its parents in the chosen DAG. There are two main methods, asymptotically equivalent, consistent and suitable to be implemented in an online manner, given a sufficient size of the training dataset, namely ML and Bayesian Estimation given Dirichlet priors [29]. In this paper, we choose ML for parameter learning, coherently with the choice of BIC as a scoring function for the structure learning algorithm. Eq. (4) represents the ML estimation.

$$\hat{\theta}_{x_i=k|parent(x_i)=j} = \frac{N_{ijk}}{N_{ij}} \cong P[x_i = k | parent(x_i) = j]. \tag{4}$$

2.2.3. Inference in BNs

The BN paradigm is mainly used to reason in domains with intrinsic uncertainty. The reasoning inside the model, that is, the propagation of evidence through the model, depends on the structure reflecting the conditional independencies between the variables. Generally speaking, the propagation of evidence involves assigning probabilities to the values of a subset of non-instantiated variables when the values of some other variables are known [25]. It has been proven that the inference in a BN with a general *DAG* structure is NP-hard [30]; however probabilistic inference algorithms that are more efficient than the brute force use of the gigantic joint probability table, have been developed by exploring the interdependency captured by the network structure. Most important among them are algorithms for computing posterior probabilities Pr $(x_i \mid e)$, where e denotes evidence, the observed values for some variables. These class of algorithms include "belief propagation" [30–31] and "Junction tree" [32–33] for exact solutions, and various statistical sampling techniques (e.g., Markov Chain Monte Carlo sampling) for approximate solutions with extremely large BN (see [34] for a detailed explanation of the most commonly used BN inference algorithms).

2.3. Learning automata

In this section, learning automata [35–36] 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 action by repeated interactions with a random environment. In this case, the environment evaluates this taken action and responds by a reinforcement

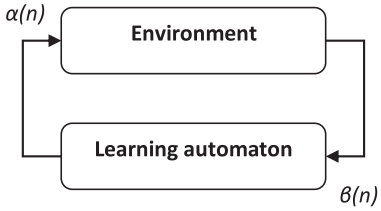

Fig. 1. The interaction between learning automaton and the environment.

signal. The learning automaton then updates its internal information, which is in the form of a probability vector, according to both the chosen action and received reinforcement signal. Subsequently, 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 output, 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, which will be used in this paper, with variable structure is represented by $\{\alpha, \beta, p, T\}$, where $\alpha = \{\alpha_1, \alpha_2,\ \ldots,\ \alpha_r\}$ is the 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. (5).

$$
\begin{aligned}
&p_i(n+1) = \ p_i(n) + a[1 - p_i(n)] \\
&p_j(n+1) = \ p_j(n) - ap_j(n) \quad \forall j,\ j \neq i \\
&\textit{If the received signal is taken as desirable.} \\
&p_i(n+1) = (1-b)p_i(n) \\
&p_j(n+1) = \frac{b}{r-1} + [(1-b)\ p_j(n)]\ \forall j,\ j \neq i \\
&\textit{If the received signal is taken as undesirable.}
\end{aligned}
\tag{5}
$$

The parameters a and b, in Eq. (5), denote reward and penalty parameters, respectively. If a and b are set equal, the algorithm is referred to as $L_{R\text{-}P}$; whereas, if b is very much smaller than a, the algorithm is called $L_{R\varepsilon P}$; finally, if $b = 0$, then the algorithm is $L_{R\text{-}I}$.

For a more complete description of learning automata, we have to say that, 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 [34]. Let A_1, ..., 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, which 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 [15].

3. Proposed reasoning and learning model for Cognitive Wireless Sensor Networks

In this section, a new reasoning and learning model is proposed, which uses a BN to represent the relationship between the network parameters, and performs an efficient probabilistic reasoning. As mentioned before, the problem of finding the optimal BN corresponding to the variables dependencies belongs to the class of NP-hard problems, and typically greedy algorithms are used to generate BNs. Firstly, we propose a new BN structure learning algorithm, which uses a team of learning automata and a training dataset. Training dataset contains samples of network protocols, which are gathered from previous observations. The constructed BN represents the dependence relationships among the network parameters contain both controllable and observable parameters. Then, after parameter training, which defines the numerical parameters (conditional probabilities) for the BN, BN inference is used to tune the controllable parameters of the network based on the desired values of the observable parameters. Finally, during the network uptime, gathered new training samples from new observations of the communication network and its environment, are used to update the BN. The pseudo code of the proposed algorithm is given in Algorithm 1.

As it can be seen in the pseudo code, the proposed reasoning and learning model has two phases: *BN structure and parameter learning* and *reasoning and learning*. In the following we describe these two phase.

3.1. BN structure learning[1]

As mentioned in Section 2.2.1, BN structure learning based on search-and-score approach has two components: a scoring metric and a search procedure. The proposed algorithm uses the *BIC* scoring metric, given in Eq. (3) and a team of learning automata to search among the possible structures. In what follows, we describe the search procedure [15].

Let n be the number of network protocols parameters including controllable and observable parameters. Since, the maximum number of undirected edges of a graph with n nodes is $m = \frac{1}{2}n \times (n-1)$, we use a team of m learning automata (A_1, A_2... A_m) with action set $\{i \rightarrow j,\ j \rightarrow i,\ no\ edge\}$ *for* $i = 1, .., n,\ j = 1, \ldots, n$ *and* $i \neq j$; each learning automata is associated to one undirected edge. Let the action of automaton A_l at time t denoted by $\alpha^l(t)$ and it determines either the corresponding edge should be appeared in the BN (in two case) or not. The action probability vector of automaton A_l (*for* $l = 1, 2 \ldots m$) at time t is denoted by $p^l(t) = (p^l_0(t), p^l_1(t), p^l_2(t))$, where $p^l_k(t)$ is the probability of choosing action α_k (for $k = 0, 1, 2$) at time t. Since there is no prior information about the variables dependencies at the first iteration, we use the uniform probability distribution to select actions; i.e. $p^l_k(t) = \frac{1}{3}$ ($for\ k = 0,\ 1,\ 2\ and\ l = 1,\ 2,\ \ldots,\ m$). After all learning automata select their actions according to their probability vectors, the selected actions are used to build the BN structure. Now, the constructed BN structure may have cycles. In order to remove cycles, the search procedure uses a modified version of Depth First Search algorithm (DFS), named colored DFS [37], with a random start node to navigate the current structure and mark all of the back edges which point from a node to one of its ancestors. After detecting all back edges, the search procedure removes all. Because

[1] In this section we have widely used the reference [15].

Algorithm 1. Pseudo code of the proposed model.

Phase I: BN structure and parameter learning
Let n be the number of network parameters including controllable and observable parameters, m be the number of LA, i.e. $m = \frac{1}{2}n(n-1)$, and for each learning automaton l, $\alpha_l = \{i \to j,\ j \to i,\ no\ edge\}$ $for\ i = 1, .., n,\ \ j = 1, \ldots, n\ and\ i \neq j$.
Let t be the iteration counter and initially set to **0** and $\boldsymbol{max_{it}}$ be the maximum number of iterations.
Set the initial actions probability distribution of each LA to be uniform.
While the number of iteration $\leq \boldsymbol{max_{it}}$
  Generate LA's actions, (α_1(t), α_2(t), ..., α_m(t));
  Build the structure of the Bayesian network based on LA's actions and remove all cycles from the constructed network;
  Using the knowledge-base of training samples, calculate the BIC_t value for current constructed BN;
  Let $\beta(t) = Max(\frac{BIC_t - BIC_{t-1}}{BIC_{max} - BIC_{min}}, 0)$;
  Update the actions probability distribution of each LA based on the reinforcement signal β(t);
Execute the ML-based parameter learning algorithm for the constructed BN;
Phase II: Reasoning and Learning
Let *new_sam* be the number of new gathered samples and initially set to **0**, and $\boldsymbol{new\_sam_{max}}$ be the maximum number of gathered samples, after which *BN structure and parameter learning* executes automatically with the extended knowledge-base of training samples.
Repeat following steps
Tune the network controllable parameters using BN inference;
Sample the network protocol parameters, both controllable and observable;
Save the values of the parameters as a new training sample in the knowledge-base;
new_sam = *new_sam* +1;
If *new_sam* = $\boldsymbol{new\_sam_{max}}$
  new_sam = 0;
  Start *BN structure and parameter learning* with the extended knowledge-base of training samples;
Else Evaluate the network performance, as ***per(t)***;
  If a drastic decrease in ***per(t)*** is observed
    new_sam = 0;
    Start *BN structure and parameter learning* with the extended knowledge-base of training samples;
Else Execute the ML-based parameter learning algorithm for the BN to update the conditional probability tables based on the updated training sample dataset;

of choosing a random node in the colored DFS, the proposed algorithm has the chance of removing different edges in the execution of the algorithm to choose the structure of the network without biasing to the any edge in the network [15].

In the next step, the reinforcement signal is computed based on the scoring functions. *BIC* given in Eq. (3) is computed for the current BN as BIC_t. Then the reinforcement signal for each automaton is calculated based on the network's score. Since we use a team of learning automata with a common payoff, all the automata get the same reinforcement signal from the environment (that is, $\beta^l(t) = \beta(t)$). The reinforcement signal of the network at t, $\beta(t)$, is computed using Eq. (6) [15].

$$\beta(t) = Max\left(\frac{BIC_t - BIC_{t-1}}{BIC_{max} - BIC_{min}}, 0\right), \tag{6}$$

where, BIC_t and BIC_{t-1} are the scores of the network in current and previous iterations, respectively. BIC_{max} and BIC_{min} also are the maximum and the minimum *BIC* values gained up to the current iteration, t, respectively. Using the BIC_{max} in calculating reinforcement signal, learning automata have more chance to escape from local optima and find better structure. Then the algorithm updates the action probability distribution of all automata using following equation:

$$p^l(t+1) = p^l(t) + \lambda\beta(t)\left(e^l_k - p^l(t)\right)\ \ l = 1, \ldots,\ \ m, \tag{7}$$

where $0 < \lambda < 1$ is a constant learning parameter and e^l_k is a unit vector of appropriate dimension, here is three, with kth component unity. It is assumed that $\beta(t) \in [0, 1]$, and this learning algorithm is known as Linear Reward-Inaction (L_{R-I}) algorithm [15].

All above steps should be repeated for max_{it} iterations.

3.1.1. A simple improvement

Since BN learning, especially structure learning, is the most computationally demanding phase in the proposed model, here, an improvement is presented to speed up the convergence.

The slow convergence rate of learning automata is a factor, which limits their convergence. In order to increase the convergence speed, a new class of learning automata, called *pursuit learning automata*, is introduced in [38], which uses an estimator algorithm. The main feature of the *pursuit learning automata* is that they maintain running estimates for the reward probability of each possible action, and use them in the probability updating equations. In *pursuit learning automata*, at each iteration, one action is selected based on its probability distribution, and then the environment responds to this action. Based on this response, the reward estimation of the action is updated. The learning automata then, pursue the action with maximal reward estimation. Changing the action probability distribution is based on both the feedback received from the environment and estimation of reward probability, using Eq. (8).

$$Q(t+1) = T(Q(t),\ \alpha(t), \beta(t)), \tag{8}$$

where $Q(t)$ is the pair of ($p(t)$, $d(t)$), and $d(t)$ is the reward probability estimator vector.

Using this class of learning automata instead of classic learning automata will increase the convergence speed of the BN learning algorithm. To do this, Eq. (7) is replaced with Eq. (9).

$$p^l(t+1) = p^l(t) + \lambda(1-\beta(t))p^l(t) + \lambda(1-\beta(t))e^l_{ma}\ \ l = 1, \ldots,\ \ m, \tag{9}$$

where *ma* is the index of the maximal component of the reward probability estimate vector $d(t)$ so that $d^l_{ma} = \max_{i=1,2,3}\{d_i(t)\}$, and e^l_{ma} is a unit vector as $[0\ 0\ 0 \ldots 1 \ldots 0\ 0\ 0]^T$, with the value of 1 being the position of the optimal action with the maximum reward estimation for A_l. According to the Eq. (9), the probability of the chosen action increases if and only if the reward estimation of it is maximal, i.e. the current action is optimal. The performance of both proposed learning automata based algorithms will be evaluated and compared in Section 4.

3.1.2. Time complexity analysis

Lemma 1. *If the number of iterations is iters and the number of random variables is n, the time complexity of the proposed BN structure learning algorithm will be* $O(iter \times n^2)$.

Proof. In each repetition, learning automata build a primary network by choosing an action from their action set; time complexity

of this step is $O(1)$. Then, DFS algorithm is used to detect and remove possible cycles in the network. The time complexity of DFS is $O(n+|E|)$, in which $|E|$ would be $\frac{1}{2}n \times (n-1)$, in the worst case; therefore the upper bound on its time complexity is, $O(n^2)$. Besides, the time complexity of computing the BIC is $O(Mn^2)$, where M is the number of training samples; and the time complexity of updating the action probability vectors for learning automata is $O(3)$ (each action probability vector has three members, whose probabilities are updated) . Finally, due to the number of iterations, *iters*, the time complexity of the proposed algorithm, may be $O(iter \times (1+n^2+Mn^2+3))$; which should be rewritten as: $O(iter \times n^2)$.

3.1.3. Convergence analysis

However in BN structure learning maximizing the expected value of the common payoff β, which depends on the *BIC* value, is the main goal of the team of learning automata; reaching a *Nash equilibrium* is another important goal in a game. Please refer to [39], to find more discussion on the rationality of Nash equilibrium. Generally, if $S^j=\{\alpha_1^j, \alpha_2^j, \alpha_3^j\}$ is the action set of automaton A_j for $1 \le j \le m$ and $S=\prod_{j=1}^{m} S_j$, for any $\boldsymbol{\alpha}=(\alpha_1, \ldots, \alpha_m) \in S$, we define its neighborhood in S as $N(\boldsymbol{\alpha})=\{x \in S | \exists i,\ s.t.\ x_j=\alpha_j,\ \forall j \neq i,\ and\ x_i \neq \alpha_i\}$. Therefore, the neighbors of any m-tuple of actions are the set of all action choices, which differ only in one action. A Nash equilibrium for this game with a common payoff is a specific m-tuple of actions $\boldsymbol{\alpha} \in S$ if and only if $\beta(\boldsymbol{\alpha}) \ge \beta(x)$, $\forall x \in N(\alpha)$, and the corresponding β, which is the highest, is called β_{opt}. Here, It should be proven that, using sufficiently small value for the step size parameter [39], the automata team would converge to one of the Nash equilibrium, if all the automata use L_{R-I}.[2]

Theorem 1. *With sufficiently small value for the step size parameter, the team of learning automata, which uses L_{R-I} as learning algorithm, would converge to one of the Nash equilibrium.*

Proof. The state of the team of learning automata at instant t is denoted by $P(t)$, with $P(t)=[P_1, P_2, \ldots, P_m]$ being the probability vector of automata; where m is the number of required learning automata for constructing a BN with n nodes, and it is calculated in this way: $m=\frac{1}{2}n \times (n-1)$. The proposed learning algorithm determines a random differential equation, which governs $P(t)$ as follows.

$$P(t+1)=P(t)+b\ G(p(t), \psi(t)), \tag{10}$$

where $\psi(t)=(\alpha(t), \beta(t))$. Moreover, $G(.,.)$ is the updating function for the jth learning automaton for $1 \le j \le m$, which uses L_{R-I} algorithm, and b is the learning parameter λ in Eq. (7) (please note that here we use j instead of l to indicate each learning automaton).

$$P_j(t+1)=P_j(t)+\lambda\beta_j(t)[e_{jq}-P_j(t)],$$

where, e_q is a vector with dimension r, which is the number of actions in the automata and its qth element being unity, if $\alpha(t)=\alpha_q$ is the chosen action of the jth automaton at instant t.

The analysis of the long time behavior of $P(t)$ proceeds in two steps: approximating the L_{R-I} algorithm using ordinary equation (ODE), and proving that the stable points in the ODE are the equilibrium points of the algorithm. In the first step, we obtain an ODE that approximates the learning algorithm of the team of automata. The general technique for this purpose is explained in [40]. In the second phase, we proved that the learning algorithm of the team of learning automata can be approximated using the ODE, actually, we want to show that if ODE has local asymptotic stable points, each $P_j(t)$ converges these equilibrium points at high values of t.

[2] However some simple convergence analysis can be found in [40], in this section we have presented it more precisely and completely.

Lemma 2. (*Approximating the L_{R-I} algorithm using ODE*) *For any initial condition P_0 and for ε, $\delta > 0$, and $\lambda^* > 0$, such that for all $0 \le \lambda^* \le \lambda$, then $prob[sub_{0 \le t \le T} \| P_t^{\lambda} - \omega(t\lambda) \| \ge \varepsilon] \le \delta$ [40].*

Proof. To approximate Eq. (10) more efficiently, we can rewrite it as follows,

$$P(t+1)=P(t)+b\ \omega(p(t))+b\theta_t, \tag{11}$$

where $\omega(P(t))=E[G(P(t),\ \psi(t))|P(t)=p]$ and $\theta_t=G(P(t),\ \psi(t))-\omega(P(t))$. Now, we will prove that Eq. (11) can be approximated using ODE in Eq. (12).

$$\frac{dg_{jq}}{dt}=\omega_{jq}^{p}(\tilde{P})\ \ for\ 1 \le q \le r_j,\ \ 1 \le j \le m, \tag{12}$$

where $\tilde{P}=[P_1, P_2, \ldots, P_t]$.

In order to show that Eq. (11) is approximated using ODE, four assumptions (A1-A4) should be held in Eq. (11).

A1. $\{P_t^b, \psi_t^b\}$ is a Markov process.

Since the algorithm is simple and repetitive, this assumption holds in Eq. (11).

A2. for any appropriate Borel set B, $prob[\psi_t^b \in B | P_t^b, \psi_{t-1}^b] = prob[\psi_t^b \in B | P_t^b]$.

The value of ψ_t, which is independent of ψ_{t-1}, is determined completely and randomly by P_t. In other words, the probability distribution $(\alpha(t), \beta(t))$ is independent of $(\alpha(t-1), \beta(t-1))$.

A3. $G(P)$: $R^N \to R^N$ is globally Lipchitz.

Let $\omega(P)=E[G(P_t, \psi_t)|P_t=P]$, a function as ω is Lipchitz if $|\omega(x)-\omega(y)| \le |x-y|$.

According to Eqs. (7) and (11), $\omega(.)$ is rewritten as Eq. (13).

$$\omega(P)=\sum_{j=1}^{m} E[\lambda B_j(t)\big(e_j-P_j(t)\big)|P_j(t)=P], \tag{13}$$

where $1 \le j \le m$ denotes the number of learning automata.

$$\omega(P)=\sum_{j=1}^{m}\sum_{q=1}^{r} E\Big[B_j(t)\big(e_{jq}-P_j(t)\big)|P_j(t)=P,\ \alpha_i(t)=\alpha_{jq}\Big].p_q,$$

where r is the number of actions in the learning automata.

$$\omega(P)=\sum_{j=1}^{m}\sum_{q=1}^{r} d_{jq}\big(e_j-P_j(t)\big).p_q$$

where d_{jq} is the reward probability for qth action in the jth learning automaton.

The sth element of $\omega(.)$ is rewritten as, $\omega_s(P)=\sum_{j=1}^{m}[\sum_{q \neq s} p_{jq}p_{js}(1-d_{jq})]$.

Then, the value of $|\omega_s(x)-\omega_s(y)|$ is calculated for the probability vectors x and y:

$$|\omega_s(x)-\omega_s(y)|=\sum_{j=1}^{m}\sum_{q \neq s}|(1-d_{js})x_{js}\Delta x_{jq}+(1-d_{js})\Delta x_{js}\Delta x_{jq}|$$

where $\Delta x_s=x_s-y_s\ \forall s,\ x_s, y_s \in [0,1]$.

$$|\omega_s(x)-\omega_s(y)|=\sum_{j=1}^{m}\sum_{q \neq s}\Big|\Delta x_{jq}\Big[x_{js}\big(1-d_{js}\big)+\Delta x_{js}\big(1-d_{js}\big)\Big] + \Delta x_{js}\big(1-d_{js}\big)x_{jq}\Big|$$

$$|\omega_s(x)-\omega_s(y)|=\sum_{j=1}^{m}\sum_{q \neq s}\Big|\Delta x_{jq}y_{js}\big(1-d_{js}\big) + \Delta x_{js}\big(1-d_{js}\big)x_{jq}\Big| \quad \text{Since } x_s, y_s \in [0,1]$$

$$|\omega_s(x) - \omega_s(y)| = \sum_{j=1}^{m} \sum_{q \neq s} \left(\left|\Delta x_{jq}\right| \left(1 - d_{js}\right) + \left|\Delta x_{js}\right| \left(1 - d_{js}\right) \right) \quad \text{Since } x_s, y_s \in [0, 1]$$

If $k_j = (1 - d_{js}) \Rightarrow |\omega_s(x) - \omega_s(y)| \leq \sum_{j=1}^{m} (k.r \sum |\Delta x_{jq}|)$.

For any automaton j, we have: $|\omega_s(x) - \omega_s(y)| \leq k.r \sum |x_s - y_s| \Rightarrow \omega$ is globally Lipchitz.

A4. $\theta_t^b = G(P_t^b, \psi_t^b) - \omega(P_t^b)$.

θ_t^b is a random variable with a mean of zero. In order to approximate Eq. (11) using ODE, we should prove that the variance θ is bounded. Since the elements P and ψ are finite in the learning automata, the variance θ is bounded, and the forth assumption holds.

Now, we want to prove that if ODE has local asymptotic stable points, P_t^{λ} converges to one of these equilibrium points at large values of t, which is the **maximum** point.

Lemma 3. (*ODE stable points are equilibrium points of the algorithm*) If χ is defined as follows:

$$\chi = \bigcap_{j,q} \{ P \in \mathrm{K} | \omega_{jq}^{p}(p) = 0 \} \; 1 \leq q \leq r_j \; 1 \leq j \leq m,$$

Prove that all corners of K belong to χ.

Definition 1. According to $\mathrm{K} = \prod_{j=1}^{N} p_{rj}$ every point in K indicates a probability distribution for the actions chosen by the team of learning automata; p_{rj} is the rth action probability of the jth learning automaton.

Definition 2. According to *corner of* $K = \prod_{j=1}^{N} p_j^*$, every corner point in K is where the probability vector of each automaton (P_j) converges to a random variable (P_j^*) with a continuous distribution.

Proof: Let $P^* = [P_1{}^*, P_2{}^*, \ldots, P_m{}^*]$ be the corner points in K. Now, we should prove that $\omega_{jq}^{p}(P^*) = 0$, meaning that in the jth learning automaton, the probability vector P_j convergesto a random variable P_j^* with a continuous distribution.

Since $\omega_{jq}^{p}(p)$ approximates the vector P, the asymptotic behavior of P(t) is expressed using $y(t)$ in Eq. (14).

$$y(t) = \frac{P(t) - g(\lambda t)}{\sqrt{\lambda}} \tag{14}$$

Let P^* be the unique equilibrium state of Eq. (13) such that $\omega_{jq}^{p}(P^*) = 0$. Moreover, let this equilibrium state be globally and asymptotically stable as well. Thus, for every initial condition g(0), we have $\lim_{t \to \infty} g(t) = P^*$. If $F(y)$ indicates a random variable distribution for y, and if $N(\mu, \Sigma)$ is a normal distribution with mean μ and covariance Σ, the asymptotic behavior of $y(t)$, and consequently of P^* and P, is calculated as follows for the small values of λ:

If $x \in R^r$ *and*P*is a* $(r^* r)$ matrix such that the inner product of $(x\text{-}P^*)$ and $\omega_{jq}^{p}(x)$ satisfies the condition$[(x - P^*), \omega_{jq}^{p}(x)] < 0 \; \forall \; x \neq P^*$, then F $(y(t)) \to N[0, \Sigma]$ *as* $\lambda \to 0$ *and* $t \to \infty$.

It can be concluded that P^* is the only zero of $\omega_{jq}^{p}(P)$, indicating that this condition is sufficient for the global asymptotic stability of the equilibrium state P^*.

A conclusion that can be drawn from Lemma 3 is that the ODE solutions do not converge to the points in χ, which are not a maximum. Furthermore, all fully-pure points are locally stable. As a result, ω_{jq}^{p} is equal to zero for the corner points in K. In other words, these points are the equilibrium points of ODE. Finally, as the function is zero for these points, the team of automata will converge to one of these points, which is the **maximum** point.

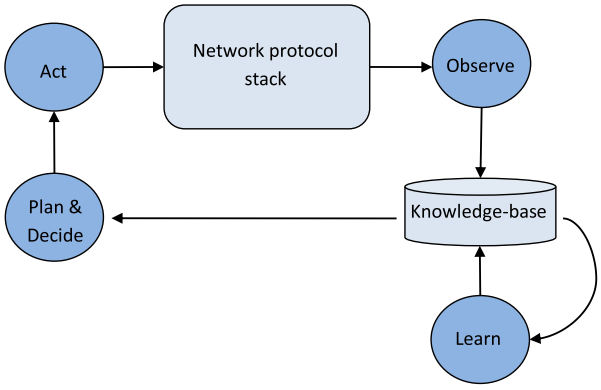

Fig. 2. Cognition cycle in the proposed cognitive network.

3.2. BN parameter learning

Once the BN structure is completed, the probabilities are entered into the network in the form of Conditional Probability Table (CPT) for each variable in the BN, to carry out the parameter learning. These tables can be constructed by learning from training dataset. For variables with no parents, this task is simple and their CPTs are reduced to their prior probabilities; however, for variables with parents, the number of entries exponentially grows with the number of patent nodes. The small tables can be filled easily, and the large tables may be break into small ones by introducing additional nodes, called *hidden nodes* [41]. In this study all of the tables are small enough to be populated by using ML estimation, which is presented in Eq. (4).

3.3. Reasoning and learning

After constructing the BN, it can be used to make various inferences about variables in the model. Nodes without parents, called roots, are often (not always) variables, whose values may be observed and cannot be changed. When no observation is made, they are characterized by their prior probabilities. The first inference is computing the prior distributions of other variables based on the priors of the roots and the CPTs of other nodes. Reasoning and learning in the cognitive networks can be studied in different steps of cognition cycle [42]. A simple cognition cycle scheme, which is used in this study, is depicted in Fig. 2. In the *plan & decide* step various controllable parameters values are determined in order to meet the performance requirements imposed by the application layer. By using Bayesian inference, the value of certain network parameters can be predicted based on the observation of some other parameters using Eq. (15) [43–44].

$$\hat{x}_i^t = E\left[x_i^t | e\right], \tag{15}$$

where $\hat{x}_i^t$ is the estimated value for x_i at time t, and e is evidence, the observed values for some other variables. Actually, the estimate of an unobserved parameter is the expectation of such parameter based on the CPT, where the probability is conditioned on the set of observed parameters.

Then in the *act* step, the network protocol stack is reconfigured based on determined values in the *plan & decide* step. Compare with other phases, this step is far less complex. In the *observe* step, the observation within the protocol stack is done by periodically sampling the observable and controllable parameters across all layers (to support a cross-layer optimization). All observations are organized and saved in a knowledge-base, as training samples of the BN, in order to be utilized for the next updates of the BN. Finally, the *learn* step consists of updating process for the BN based on the information gathered in the *observe* step. In the

learn step both structure and parameters of the BN may be updated. To do this, the *BN structure and parameter learning* phase in Algorithm 1 is executed. Since this phase (specifically structure learning) has computational overhead and causes some delay in the network, which may bother the users, it is performed as a background activity during the network active times. Also, if it is possible, it may run offline when the network is not active. As it can be seen in Algorithm 1, Phase II, lines 7–9, BN learning process automatically runs, after $\boldsymbol{new\_sam_{max}}$ new samples are added to the training dataset. The $\boldsymbol{new\_sam_{max}}$ may be adjusted based on the features of CWSN applications, such as its sensitivity to the time, rate of events, and etc. On the other hand, if the network performance has a drastic decrease, the BN learning phase immediately runs (see Algorithm 1, Phase II, lines 11–13) and a new constructed BN replaces the previous BN. Therefore, in the *learn* step, the BN is replaced by the updated one, whenever the *BN structure and parameter learning* phase completes. This always keeps the BN updated and adapted with the communication network environment. Moreover, since ML-based parameter learning procedure is simple and has low computational overhead, CPTs are regularly updated (according to Algorithm 1, Phase II, line 14), even if the BN structure is fixed. By implementing the learning phase in this way, cognition cycle will be completed which means that the performance of CWSN starting by few numbers of samples may catch up with the performance of CWSN starting by large numbers of samples over time.

3.4. CWSN scenario

Generally, WSNs and also CWSNs can be characterized by their applications or objectives that their users want to achieve; here, we try to overcome some of the challenges, which are mentioned in Section 1, in a proposed traffic management scenario. Firstly, we should define controllable and observable parameters. Five controllable parameters are considered as below: Sensing Interval (SI) in the application layer, which determines the duration of asleep and awake. Congestion Window (CgW) in the transport layer, i.e., the total amount of data [bytes] that can remains unacknowledged at any time. We suppose that the routing protocol in the routing layer is reliable and we do not consider it in this study. These parameters are effective in routing and forwarding process. In the MAC layer we consider, duty cycle (DC) to define how long a node has to wait for receiving the acknowledgment, after sending a frame; contention window (CtW), which determine the maximum number of slots a node wait before transmitting a packet according to a random back off interval between zero and contention window; and max-retransmission (MxR), which limits the number of retransmission for a lost frame. The mentioned parameters of the MAC layer are important in send/receive durations. Each controllable parameters value is chosen from a finite set of standard values, which are usually used in the configuration of WSNs by an expert.

We also consider five observable parameters as below: in the transport layer, Round Trip Time (RTT) and throughput (Thp) are considered. RTT is the time between when a packet is sent and when the corresponding acknowledgment is received. Thp is the total amount of unique data [bytes] acknowledged in a sampling interval. In the MAC layer, total number of original sent frame in a sampling interval (TS), and total number of resent frames in a sampling interval (TrS); the later depicts the number of packet-loss. Finally, power consumption (Pw) in a sampling interval is measured as a very important metric in all WSNs.

Most of the above parameters, both controllable and observable have a finite number of outcomes; also some of them, such as RTT, and CW have been quantized to n_q levels, so, we have multinomial variables, which are computationally more efficient.

After determining all the parameters, *BN structure and parameter learning* phase starts. In this phase, a BN is constructed to represent the relationship between the parameters, both controllable and observable, using an available training dataset. The training dataset has consisted of samples, which are previously stored in the network active times. Minimum required samples may be varied in different application; however, it has shown that using a training dataset with 1500 samples, the proposed BN learning algorithm performs well [15]. If there is no prior information, network starts with default parameters setting, and the training dataset is being made by adding samples, which have been gathered during the network active time. In this case, the BN learning algorithm will start, if at least 100 samples exist in the training datasets. Then, in the *reasoning and learning phase* cognition cycle, which is shown in Fig. 2, runs. As the result of *plan & decide* and *act* steps, the network will be configured, in a way that satisfies the network performance metrics. Simply, we take a centralized view of the network, i.e., all nodes use the same parameter values and all nodes evaluate their parameter in the same way. To do this, we have considered an access point, which is responsible for learning a suitable BN based on the training dataset. After constructing the BN and performing the Bayesian inference, the access point will declare the appropriate configuration to the network nodes; this consideration will leads to even more effective learning and less overhead. For large scale networks several access point can be considered, which play a game with a common payoff to find the proper configuration for the networks. In this case, the behavior of the access points should be analysis through a game theory approach. It will be considered as the writers' future work.

4. Performance evaluation

In order to evaluate the performance of the proposed model for Cognitive Wireless Sensor Network (CWSN), some simulations are conducted. Firstly, we evaluate the performance of the proposed BN learning algorithm on well-known BNs using existing datasets. Then, the performance of the proposed learning and reasoning model in a CWSN scenario will be evaluated and compared with the performance of a similar WSN without cognition in different cases. Finally, the performance of proposed CWSN has been compared against cognitive radio and cross-layer design. The communication network is simulated using ns-3 [45], discrete event network simulator and the data collected from ns-3; while, the BN learning algorithm is implemented in MATLAB.

4.1. Performance evaluation of the proposed BN structure and parameter learning algorithm

BN learning is performed by the proposed algorithm, which is presented in Section 3.1. This phase is the most computationally demanding. The performance of the proposed structure learning algorithm has been evaluated in [15], and the results have showed that using a team of learning automata is a superior solution for BN structure learning, compared against many other solutions. Here, firstly, we evaluate and compare two proposed learning automata based BN structure learning algorithms. To do this, two well-known BNs are considered and the proposed algorithms are employed to construct them. Hereinafter, the **L**earning **A**utomata based **A**lgorithm is called LA-A, and the **P**ursuit **L**earning **A**utomata based **A**lgorithm is called PLA-A. The well-known BNs, which usually are used in the experiments of BN structure learning researches, are: the ALARM [46] and the ASIA [47] networks. ALARM is a medical diagnostic alarm message system for patient monitoring; it contains 37 nodes and 46 arcs (Fig. 3(a)). ASIA is a simple network with eight binary nodes and eight arcs; it was introduced

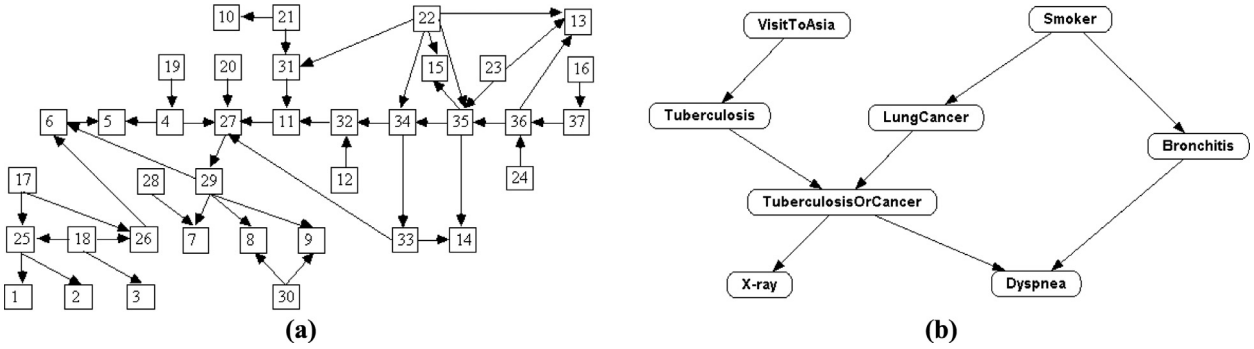

Fig. 3. (a) The ALARM network; (b) the ASIA network.

by Lauritzen and Spiegelhalter to illustrate their method of propagation of evidence, considers a small piece of factious qualitative medical knowledge (Fig. 3(b)). To generate the training datasets in these experiments, we use the 5000 first cases of a database that was generated by Edward Herskovits [48] for the ALARM, and a dataset with 5000 cases, which is sampled using the Netica tool [49] for the ASIA.

Performance metrics, which are considered, are [15]:

I. The *BIC*
II. *Hamming Distance*
III. Computation time.

BIC is the score of the constructed BN and measured by Eq. (3); its interpretation is: the higher *BIC*, the network is better. *Hamming Distance* directly compares the structure of the learned and the original networks. We define the *Hamming Distance* between two DAGs as the number of the following operators required to make the DAGs match: add or delete an undirected edge, and reverse the orientation of a directed edge; the lower *Hamming Distance* indicates the better network. LA-A and PLA-A are also evaluated based on the computation time; both algorithms are implemented in MATLAB in a PC, which has a single CPU of Intel(R) Core^{TM} 2 Duo 3.33 GHz and a 1GB memory. The algorithms run with no prior limitation in number of iterations, until more repetition does not increase the *BIC*. Table 1 represents the results; each reported statistic is the average over 20 runs of each algorithm on the training datasets.

The *Hamming Distance* in both LA-A and PLA-A indicate that the constructed networks for the ASIA are identical to its origin; but for the ALARM two arcs {21–31 and 12–32} are missed. A subsequent analysis revealed that missing arcs of the ALARM are not supported by the 5000 cases, and their nodes are actually independent in the employed database. The *BIC* and computation time show that PLA-A has had a better performance especially, in terms of computation time. These results were not far from our expectation because using the reward estimator in the pursuit learning automata speed up the convergence rate in comparison with the classic learning automata. However, since pursuit learning automata has a large memory overhead [35] and sensor nodes in WSNs have limited memories, we will continue the experiments using LA-A to build the BN of the communication network parameters. In situations, where there are no memory limitations, PLA-A will be the more appropriate choice.

In the following, using a communication network samples, we evaluate and compare the BNs, which are built using datasets that differ by the number of training samples. Structure learning algorithm is performed using datasets with 500, 1000, 2000, 3000 samples of the network parameters. The number of parameters is $N = 10$. Fig. 4 illustrate the constructed BNs; (a) the constructed BN using dataset with 500 samples; (b) the constructed BN using dataset with 1000 samples; (c) the constructed BN using dataset with 2000 samples; and finally, the constructed BN using dataset with 3000 samples.

As the Fig. 4 depicts, using training datasets with more samples causes more accurate BN structure; however, even the BN, which is built using the fewest samples, do not show wrong relations, it just represents some unnecessary relations. So, we can easily trust to the constructed BN and run the cognition cycle.

In the next step, after constructing the BN, parameter learning algorithm must define the numerical parameters (conditional probabilities) for the given network topology, using Eq. (4) [43–44]. Eq. (4) determines quantitative probability relations among the communication network parameters based on the given training datasets.

4.2. Performance evaluation of the proposed learning and reasoning model in CWSN

In this section, the performance of the proposed model in CWSN is evaluated and compared against a WSN without cognition. To do this, a BN model is constructed to represent the relations among controllable and observable parameters, which are previously introduced. As mentioned before, structure learning is done using a team of learning automata and Eq. (4) is used for parameter learning. These are carried out in *BN structure and parameter learning* phase; then *reasoning and learning* phase will start.

In the *reasoning and learning* phase, cognition cycle, which has been presented in Fig. 2, runs. In the *plan & decide* step various

Table 1
Experimental results of comparison the performance of LA-A and PLA-A on the ALARM and ASIA.

Metrics						
Algorithms	ALARM			ASIA		
	BIC	Hamming Distance	Computation time	BIC	Hamming Distance	Computation time
LA-A	0.19472	2.0	195.14	0.11494	0.0	71.12
PLA-A	0.20853	2.0	116.14	0.12145	0.0	35.5

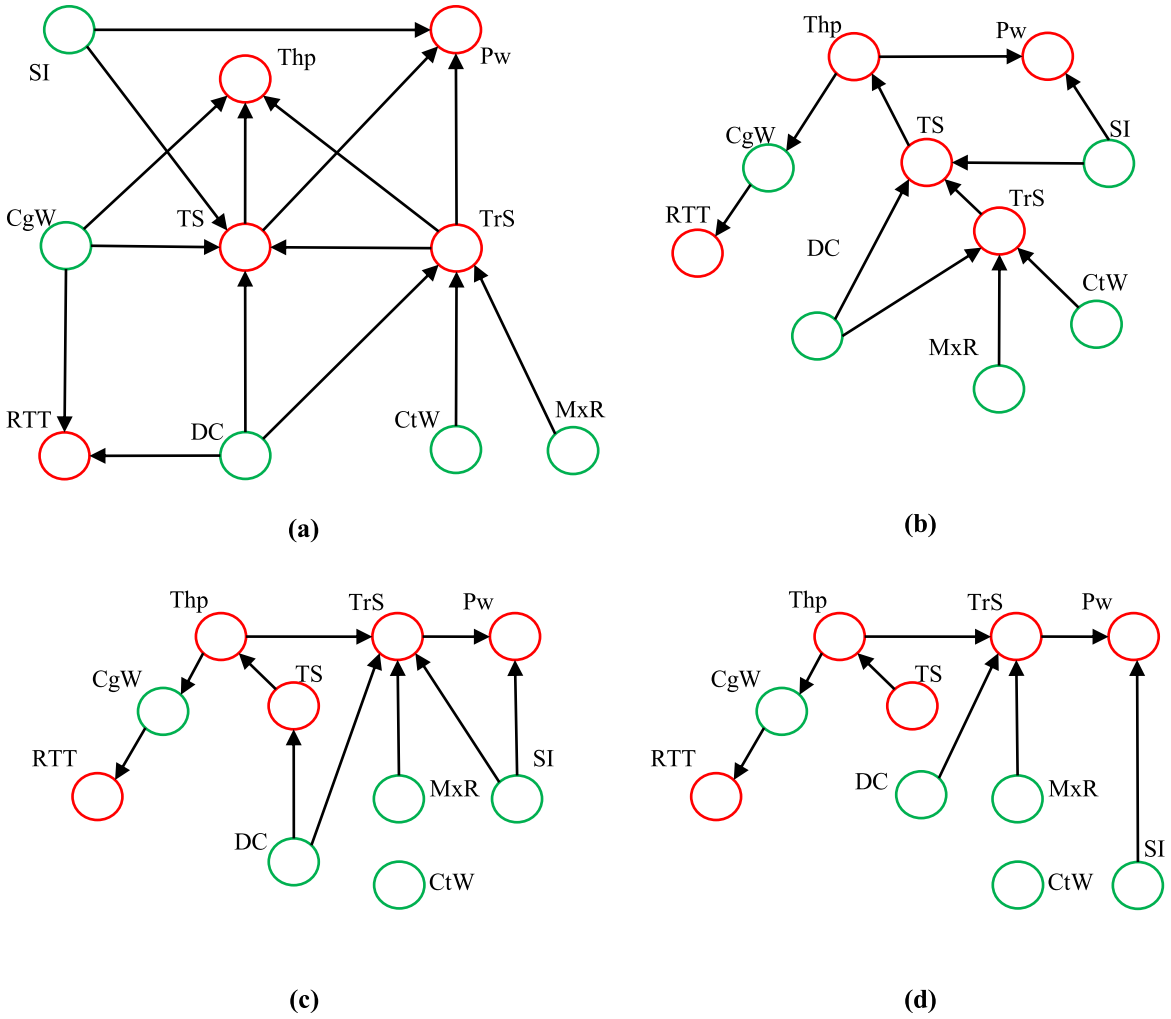

Fig. 4. The constructed BNs using training datasets with, (a) 500 samples, (b) 1000 samples, (c) 2000 samples, (d) 3000 samples.

controllable parameters values are determined by using Bayesian inference, which predicts the value of certain network parameters based on the observation of some other parameters using Eq. (8). Then, in the *act* step the parameters of the network protocol stack are set based on values defined in the *plan & decide* step. In the *observe* step the protocol stack will be monitored by periodically sampling the observable and controllable parameters, and the observations are organized and saved in a knowledge-base. Finally, the *learn* step will update the BN based on the information gathered in the *observe* step. In the *learn* step both structure and parameters of the BN may be updated.

A varied range of scenarios are considered, network sizes varied from fifty to two hundred nodes with random topologies. Network is queried every n milliseconds and get information on the nodes performance; parameters of the nodes are adjusted as result of *plan & decide* step. Data collected from the NS-3 is used, while the BN model is written in MATLAB, as mentioned before. Observable parameters, which are: Pw as power consumption, RTT to compute the average delay, TrS as the number of lost packets, and Thp as throughput, when TS, the number of original packets sent is 20,000, are measured. Finally, The utility function, which maps the observable parameters' values to a single numerical according to the deployed stack and the operational environment is considered as another evaluating metrics. In these simulations the BN, which is previously constructed using 3000 training samples, is used. Fig. 5 represents the results.

We also consider a varied range of situations, in which the CWSN will start by different training datasets, from a dataset with no training sample to datasets with 500, 1000, 2000, 3000 samples of the network parameters (as Section 4.1). The number of sensor nodes in the communication network is 200. Observable parameters are monitored during a fix interval, and samples gathered during this interval are averaged. Fig. 6 illustrate the results. Actually, this is a sensitivity analysis for studying the effect of training dataset size on the performance of the proposed CWSN. As the results illustrate, in average, the proposed CWSN always shows better performance compared with WSN; even when it starts with no prior information, i.e. with no training sample, and the corresponding BN is built during the communication network activity. For example Pw of WSN is 24.48 (Fig. 5(a)), while in the same situation Pw of CWSN, which starts with no training sample is 20.01 (Fig. 6(a)); and as for other observable parameters.

We also study the behavior of the network in the case that a drastic change happens in the network's performance and the CWSN has to react quickly and appropriately. As an example, there might be a sudden increase in the node failure rate when the network focus shifts from low-power message delivery to just message delivery, in a time-sensitive application. In the next simulations we will evaluate the utility decrease in this situation and also the time duration, when the proposed CWSN will adapt itself by the new condition. Fig. 7 depicts the results. When a drastic change happens, WSN's performance seriously decreases and it re-

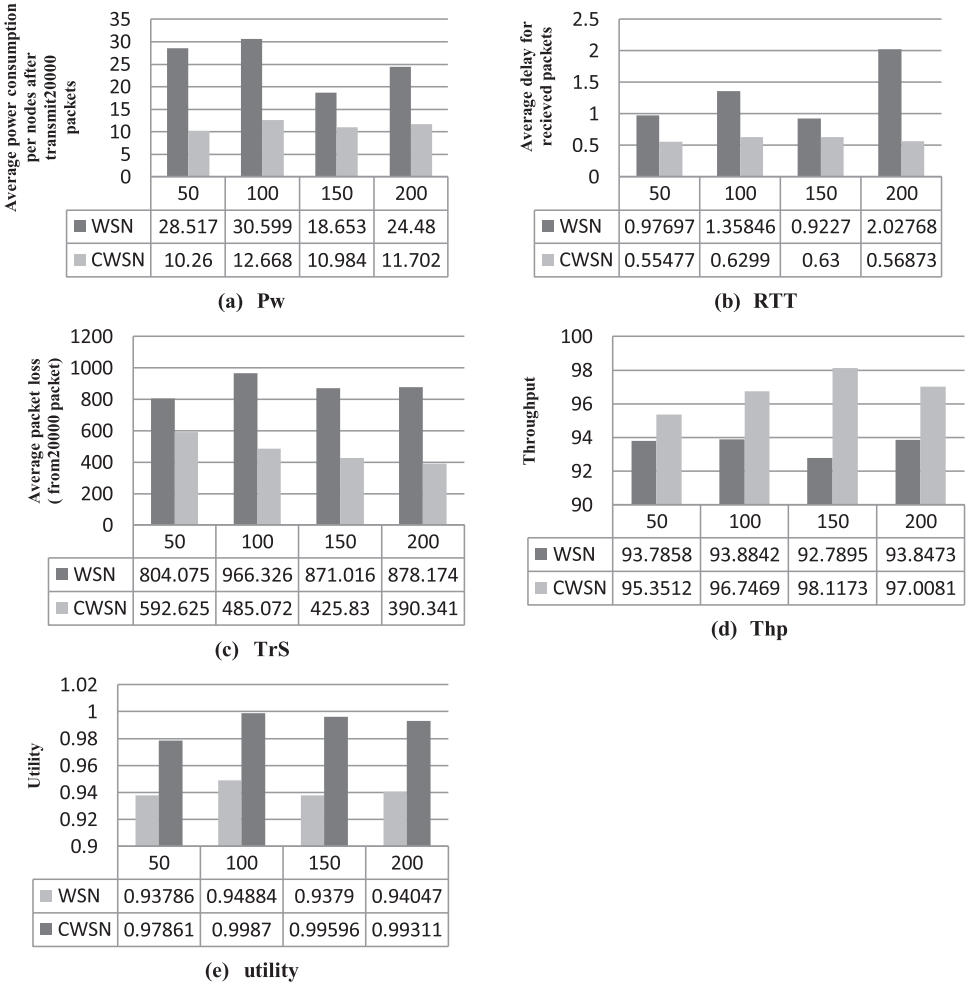

Fig. 5. Performance evaluation of the proposed CWSN based on the BN model constructed using dataset with 3000 samples, compared with simple WSN.

mains until an exterior manager reconfigures the network parameter to adapt with new situation. On the other hand, the CWSN's performance also seriously decreases when a drastic change happens; however, the proposed CWSN can adapt itself with new situations without the intervention of an exterior force, and come back to the previous conditions.

In order to adapt the controllable parameters to achieve the desired throughput, end to end delay, or the number of lost packets, we need to exploit the Bayesian inference to estimate the Thp, RTT, or TrS; so, accurate estimations is a very important issue. To measure the accuracy of the estimations, Normalized-Root-Mean-Square-Deviation (NRMSD), which facilitates the comparison between datasets or models with different scales, is used. Root-Mean-Square-Deviation (RSMD) of predicted values, $\hat{x}_t$ for times t of a regression's independent variable x is computed for n different predictions as the square root of the mean of the squares of the deviations (Eq. (16)):

$$RMSD = \sqrt{\frac{\sum_{t=1}^{n} \left(\hat{x}_i - x\right)^2}{n}}. \tag{16}$$

Though there is no consistent means of normalization in literatures, the range of the measured data defined as the maximum value minus the minimum value is a common choice, $MSD = \frac{RMSD}{x_{max} - x_{min}}$. We measure the accuracy of the Thp, RTT, and TrS estimated values using the BNs were depicted in Fig. 4, and Eq. (15), where all the parameters except, $\overline{\text{Thp}}$ for estimating the Thp, $\overline{\text{RTT}}$ for estimating the RTT, and $\overline{\text{TrS}}$ for estimating the TrS are observed as the evidence e. Fig. 8 shows the results.

As expected and Fig. 8 shows, although BNs, which have been trained using datasets with more cases have better performances at the start time; but BNs, which have been constructed using dataset with small samples achieve almost the same performance over the time. Fig. 8(a) depicts that approximately after 9 time units, NRMSDs of throughput for four BNs, which have been constructed using datasets with different number of samples, become nearly similar. Also, the differences between NRMSDs for the number of lost packets (Fig. 8(b)) and end to end delay (Fig. 8(c)) between the BNs, after 12 time units, are 0.07 and 0.06, respectively, in the worst case. Overall, this experiment shows that the performance of CWSN starting by even 500 samples may catch up with the performance of CWSN starting by 1000, 2000, or even 3000 samples with the passage of time.

Finally, observing the performance results, we conclude that the Bayesian inference helps the *plan & decide* step in predicting the behavior of certain network parameters and also in guaranteeing the quality of Thp, RTT, and TrS estimates with proper BN topology.

4.3. Compare the performance of the proposed CWSN against cognitive radio and cross-layer design

As mentioned in introduction, cognitive network, which is our novel contribution, is more than cognitive radio (in physical layer) and cross-layer design. Therefore, here, we compare the perfor-

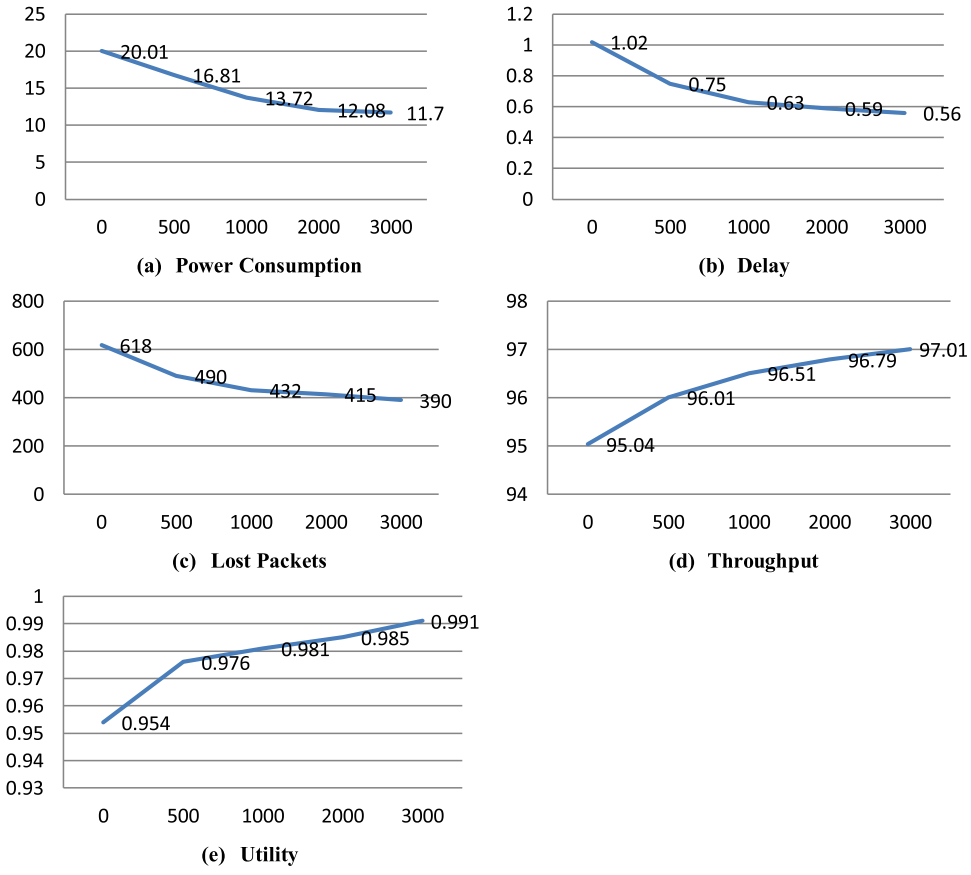

Fig. 6. Performance evaluation of the proposed model of CWSN when the BN is built using different size datasets.

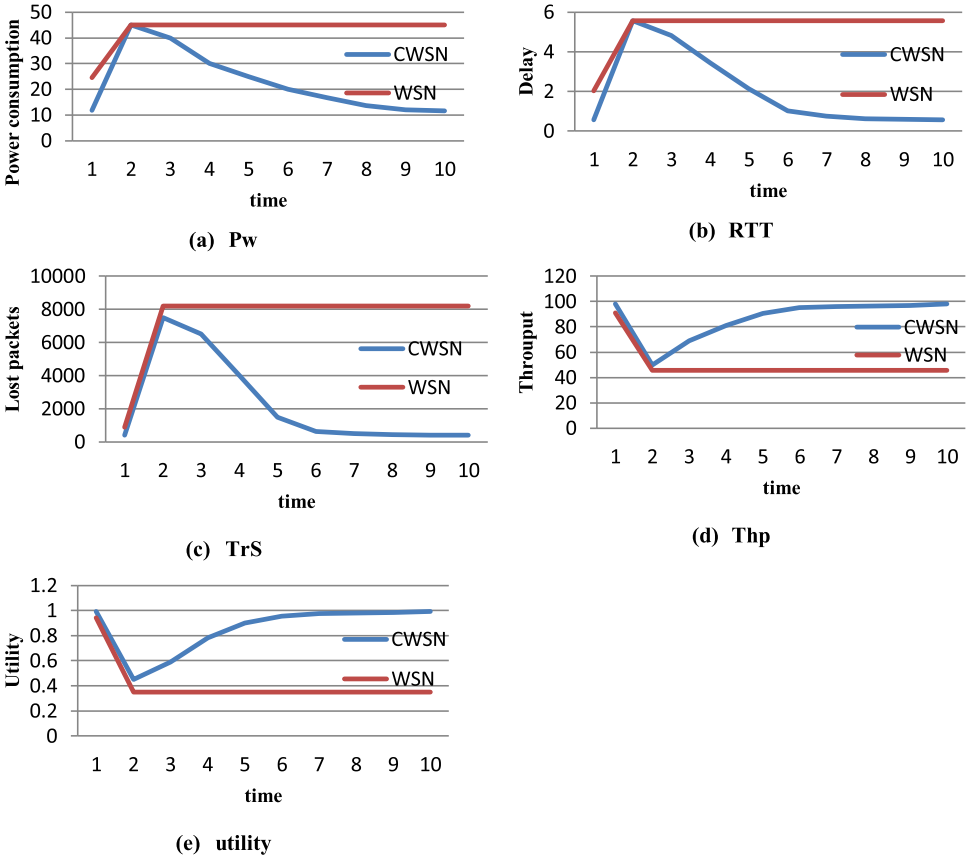

Fig. 7. The behavior of WSN and CWSN when a drastic change happens in the network's situation.

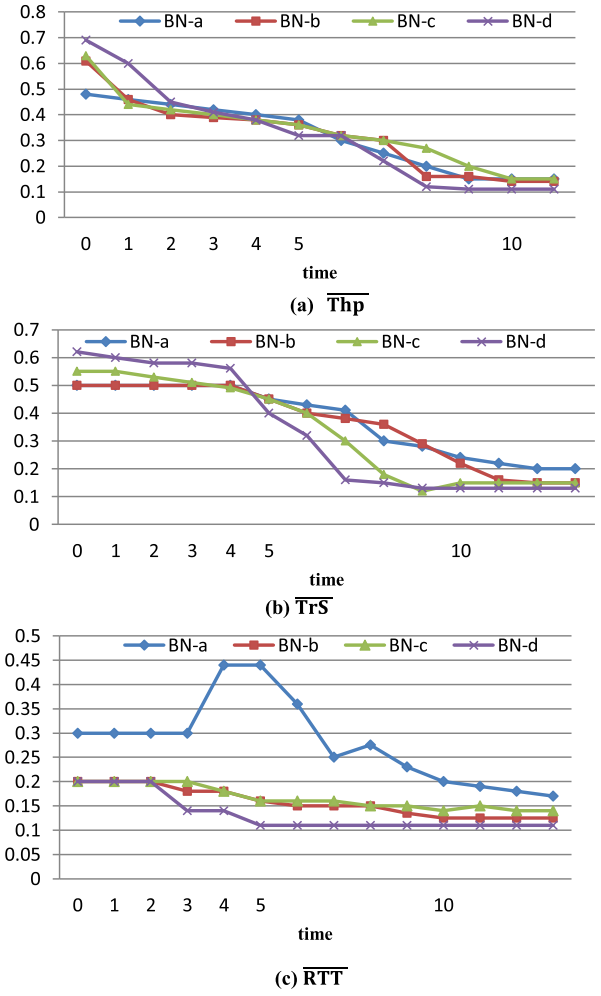

Fig. 8. The NRMSD of the Bayesian inference for estimating, (a) throughput, (b) the number of lost packets, and (c) end to end delay for the constructed BNs, which are shown in Fig. 4.

mance of CWSN against cognitive radio-based WSN (CR-WSN) and a cross-layer protocol for WSN to show that the CWSN can gain much improvement.

There are many articles, which have focused on using cognitive radio technology in WSN such as [50–56]. In all these papers, authors have applied Dynamic Spectrum Access (DSA) model in WSNs to get access to less congested spectrum, possibly with better propagation characteristics. Also, there are some researches, which have investigated cross-layer design for WSNs such as [57–66]. In these studies different cross-layer protocols have been developed and they have replaced the entire traditional layered protocol architecture that has been used so far in WSNs.

As expected, CR-WSN has usually introduced new features incorporated into the WSN physical and MAC layer protocols to support operation as a secondary system according to a DSA model [51]. Therefore, in researches, the basic 802.11.4 standard, which has been used for a number of WSN applications, has been considered, and the CR features that could be incorporated into 802.15.4 physical and MAC layers to operate according to the DSA model, have been described. We have let the configuration of other layers in network protocol stack unchanged and like WSN in previous Section (4.2). Please note that, however on time channel allocation to primary users is a key feature in CRs [2], we cannot consider this feature in our comparisons; because in CWSN and cross-layer design primary and secondary users are not defined. Therefore, only general metrics have been evaluated in this section.

To compare CWSN against cross-layer design, we must select a protocol, which has been proposed for a scenario similar to ours (traffic management scenario), such as the unified cross-layer module (XLM) proposed in [57] and CLB-routing in [66]. XLM melts common protocol layer functionalities into a cross-layer module for resource constraint sensor nodes. The operation of the XLM is devised based on the new notion of initiative determination, which constitutes the core of the XLM and implicitly incorporates the intrinsic communication functionalities required for successful communication in WSN. Based on the initial concept, XLM performs received based contention, local congestion control, and distributed duty cycle operation in order to realize efficient and reliable communication in WSN. Analytical performance evaluation and simulation experiment results have shown that XLM significantly outperforms the traditional layered protocol architectures in terms of both network performance and implementation complexity.

In the following, we compare the performance of CWSN against CR-WSN and XLM. 200 nodes have randomly deployed in 100 × 100 m^2 sensor field and we consider the behavior of the networks during the time. We also study the behavior of the networks in the case that a drastic change happens in the networks' performance. Fig. 9 shows the results.

Simulation results show that, generally, CWSN and cross-layer design begin with almost similar performance; but CR-WSN is less efficient. This is justified, because CWSN and cross-layer design consider and configure the entire protocol stack to optimize network-wide performance goals; while, CR-WSN applies cognition only at the physical layer to dynamically detect and use spectrum holes, focusing strictly on DSA. Therefore, since the goals of CR-WSN are restricted to physical layer, it cannot satisfy the end-to-end goals such as throughput, and power consumption.

Then, at $t = 4$ a drastic decrease happens in the performance of the network; for example, because an extreme increase in the network load occurs, or the focus of the network shifts from low-power message delivery to just message delivery, in a time-sensitive application. In this situation, networks must quickly adapt themselves with new conditions and try to increase their performance. As it is expected, CWSN can adapt itself with new condition, due to the fact that, there is a cognition cycle in CWSN. The cognition cycle can observe the network condition, and based on that, it decides and acts; it can also learn from the past. Actually, when a drastic change happens in the performance, CWSN runs BN learning phase based on the extended knowledge-base of training samples (please refer to Algorithm 1). Quickly and appropriately, necessary changes are applied to the BN; however this may take time, CWSN can increase the performance faster than other cognitive approach (CR-WSN). CR-WSN also has a cognition cycle and it can adapt itself to the network condition; but since it focuses only on the physical layer, final performance of the network again is not satisfactory. Beside these, the cross-layer design is based on an algorithmic approach and does not learn and adapt according to the end-to-end communication goals; as a result, the performance of the network will be low until the network conditions return to the previous state, or protocol settings to be updated by a supervisor. It can be seen in Fig. 9 that at $t = 4$ the performance of the cross-layer protocol sharply reduces and even with the passage of time, it does not improve.

5. Conclusion

In this study, a new reasoning and learning model is proposed to convert a Wireless Sensor Network (WSN) to a Cognitive Wireless Sensor Network (CWSN). In the proposed model a team of learning automata is employed to build a Bayesian Network (BN) of

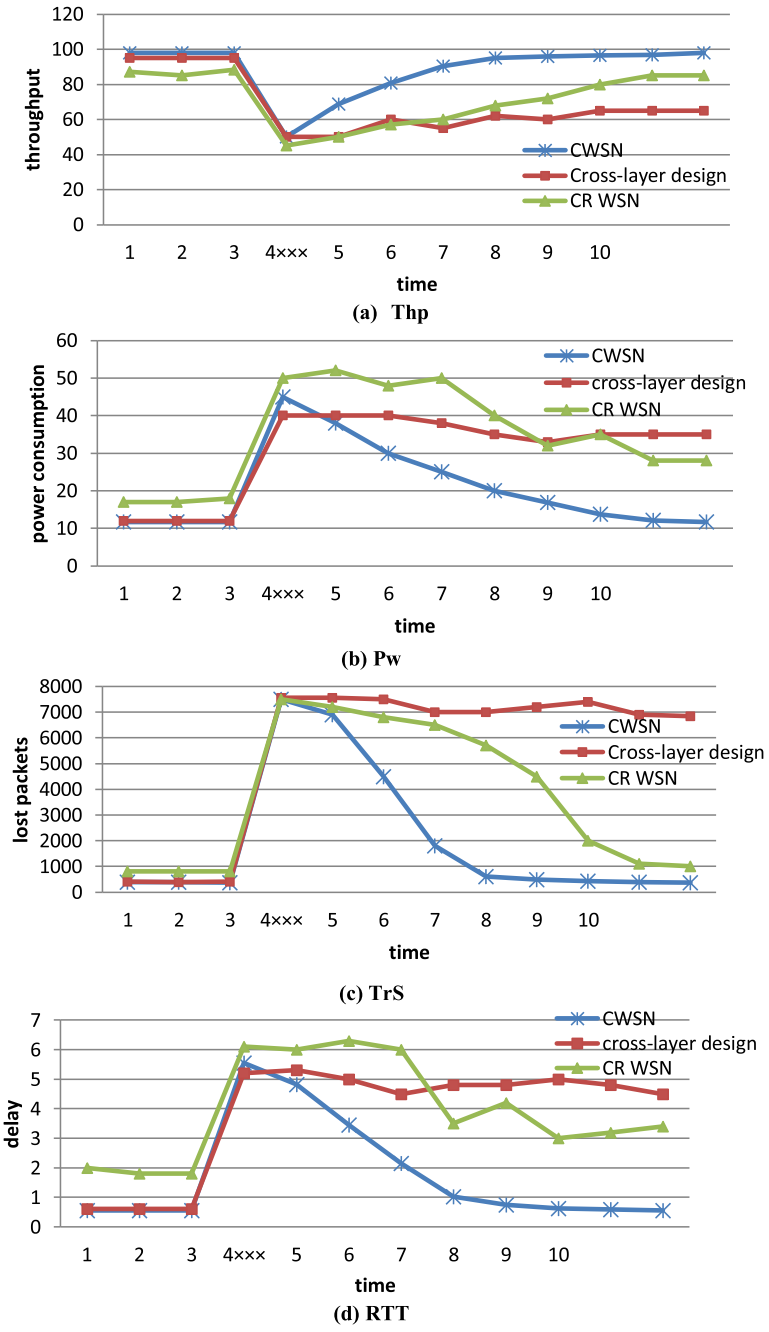

Fig. 9. Compare the performance of the proposed CWSN against cognitive radio and cross-layer design.

network parameters, which are divided to controllable and observable parameters. The constructed BN represents the independence assumptions that must be held between these parameters; then, parameter learning determines the numerical parameters or conditional probabilities for the given topology. The BN is built in the first phase of the algorithm, and in the second phase, the cognition cycle is performed. The Bayesian inference is used in *plan & decide* step for a cross-layer optimization, which considers different parameters from different protocols in the network protocol stack, to maximize the perceived network performance. After applying the decisions on the network, network performance is monitored. The observed data is used to evaluate the decisions made in previous steps and to update the constructed BN in *learn* step. Defining the *learn* step, the BN learning continues during the network life time; therefore, the BN is kept up-to-date with the communication network environment. A CWSN scenario is considered and the proposed reasoning and learning model is tested in varied situations. The results show the acceptable performance in all cases. In the future work, authors yearn to apply Influence Diagrams (*ID*), which extend the BNs with two additional nodes, namely *decision nodes* and *utility nodes* for solving decision problems, instead of the BNs; in order to have a more effective reasoning and learning model.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Netw. 38 (4) (2002) 393–422.
[2] J.M. Iii, An integrated agent architecture for software defined radio, Doctoral Dissertation, Royal Institute of Technology, Stockholm, Sweden, 2000.
[3] V.T. Raisinghani, S. Iyer, Cross-layer design optimizations in wireless protocol stacks, Comput. Commun. 27 (8) (2004) 720–724.

[4] R.W. Thomas, D.H. Friend, L.A. DaSilva, A.B. MacKenzie, Cognitive Networks, Springer, Netherlands, 2007.
[5] C. Fortuna, M. Mohorcic, Trends in the development of communication networks: cognitive networks, Comput. Netw. 53 (9) (2009) 1354–1376.
[6] T.G. Dietterich, Learning and Reasoning, Oregon State University, 2004 Tech. Rep.
[7] Y. Xiang, Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach, Cambridge University Press, 2002.
[8] C. Barrett, A. Marathe, M.V. Marathe, M. Drozda, Characterizing the interaction between routing and MAC protocols in ad-hoc networks, in: Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing, ACM, 2002, pp. 92–103.
[9] W. Watthayu, Y. Peng, A Bayesian network based framework for multi-criteria decision making, in: Proceedings of the 17th International Conference on Multiple Criteria Decision Analysis, 2004, pp. 6–11.
[10] W. Buntine, Theory refinement on Bayesian networks, in: Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc, 1991, pp. 52–60.
[11] W. Lam, A.M. Segre, A distributed learning algorithm for Bayesian inference networks, Knowl. Data Eng. IEEE Trans. 14 (1) (2002) 93–105.
[12] L.M. De Campos, J.M. Fernandez-Luna, J.A. Gámez, J.M. Puerta, Ant colony optimization for learning Bayesian networks, Int. J. Approximate Reasoning 31 (3) (2002) 291–311.
[13] J.I. Alonso-Barba, J.A. Gámez, J.M. Puerta, Scaling up the greedy equivalence search algorithm by constraining the search space of equivalence classes, Int. J. Approximate Reasoning 54 (4) (2013) 429–451.
[14] J.W. Myers, K.B. Laskey, K.A. DeJong, Learning Bayesian networks from incomplete data using evolutionary algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference, 1, 1999, pp. 458–465.
[15] S. Gheisari, M.R. Meybodi, M. Dehghan, M.M. Ebadzadeh, Bayesian network structure training based on a game of learning automata, Int. J. Mach. Learn. Cybern. (2016) 1–13.
[16] S. Gheisari, M.R. Meybodi, M. Dehghan, M.M. Ebadzadeh, BNC-VLA: Bayesian network structure learning using a team of variable-action set learning automata, Appl. Intell. (2016) 1–17.
[17] P. Larranaga, C.M.H. Kuijpers, R.H. Murga, Y. Yurramendi, Learning Bayesian network structures by searching for the best ordering with genetic algorithms, IEEE Trans. Syst. Man Cybern. 26 (1996) 487–493.
[18] H. Steck, Constraint-based structural learning in Bayesian networks using finite data sets, Technical University of Munich, Department of Computer Science, PhD Thesis (2001).
[19] Cheng, Jie, David Bell, and Weiru Liu. "Learning Bayesian networks from data: an efficient approach based on information theory." Technical report, University of Alberta, Canada, 1998.
[20] B. Moradabadi, H. Beigy, A new real-coded Bayesian optimization algorithm based on a team of learning automata for continuous optimization, Genet. Program. Evolvable Mach. 15 (2) (2014) 169–193.
[21] I. Tsamardinos, L.E. Brown, C.F. Aliferis, The max-min Hill-climbing Bayesian network structure learning algorithm, Mach. Learn. 65 (1) (2006) 31–78.
[22] C. Yuan, B. Malone, X. Wu, Learning optimal Bayesian networks using A* search, in: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, 22, 2011, p. 2186.
[23] M. Pelikan, D.E. Goldberg, Bayesian Optimization Algorithm: from Single Level to Hierarchy, University of Illinois at Urbana-Champaign,, Champaign, IL, 2002.
[24] D. Heckerman, A tutorial on learning with Bayesian networks, in: Innovations in Bayesian Networks, Springer, Berlin Heidelberg, 2008, pp. 33–82.
[25] P. Larrañaga, H. Karshenas, C. Bielza, R. Santana, A review on evolutionary algorithms in Bayesian network learning and inference tasks, Inf. Sci. 233 (2013) 109–125.
[26] D.M. Chickering, D. Geiger, D. Heckerman, Learning Bayesian Network is NP-Hard, 1994 Technical Report MSR-TR-94-14.
[27] D. Heckerman, A tutorial on learning with Bayesian networks, in: Innovations in Bayesian Networks, Springer, Berlin Heidelberg, 2008, pp. 33–82.
[28] M. Pelikan, D.E. Goldberg, Bayesian Optimization Algorithm: From Single Level to Hierarchy, University of Illinois at Urbana-Champaign, Champaign, IL, 2002.
[29] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT press, 2009.
[30] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artif. Intell. 42 (2) (1990) 393–405.
[31] H.E. Kyburg, Probabilistic reasoning in intelligent systems: networks of plausible inference by Judea Pearl, J. Philos. 88 (8) (1991) 434–437.
[32] F.V. Jensen, K.G. Olesen, S.K. Andersen, An algebra of Bayesian belief universes for knowledge-based systems, Networks 20 (5) (1990) 637–659.
[33] A.L. Madsen, F.V. Jensen, Lazy propagation in junction trees, in: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 362–369.
[34] E. Castillo, J.M. Gutierrez, A.S. Hadi, Expert Systems and Probabilistic Network Models, Springer Science & Business Media, 2012.
[35] M. Thathachar, P. Shanti Sastry, Varieties of learning automata: an overview, Syst. Man Cybern. Part B 32 (6) (2002) 711–722.
[36] K.S. Narendra, M.A. Thathachar, Learning Automata: An Introduction, Courier Corporation, 2012.
[37] Amir Kamil, "*Graph Algorithms*", CS61B, University of California, Berkeley, 2003.
[38] M.A.L. Thathachar, P.S. Sastry, A class of rapidly converging algorithms for learning automata, IEEE Trans. Syst. Man Cybern. (1985) 168–175 SMC-15.
[39] P.S. Sastry, V.V. Phansalkar, M.A.L. Thathachar, Decentralized learning of nash equilibria in multi-person stochastic games with incomplete information, IEEE Trans. Syst. Man Cybern. 24 (1994) 769–777.
[40] M.A. Thathachar, P.S. Sastry, Networks of Learning Automata: Techniques for Online Stochastic Optimization, Springer Science & Business Media, 2011.
[41] R.E. Neapolitan, Learning Bayesian Networks Pearson, Prentice Hall, Upper Saddle River, New Jersey, 2004.
[42] Q.H. Mahmoud, in: Cognitive Networks, John Wiley& Sons Ltd, 2007, pp. 57–71.
[43] G. Quer, H. Meenakshisundaram, B. Tamma, B.S. Manoj, R. Rao, M. Zorzi, Cognitive network inference through Bayesian network analysis, in: *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, IEEE, 2010, pp. 1–6.
[44] G. Quer, H. Meenakshisundaram, B.R. Tamma, B.S. Manoj, R. Rao, M. Zorzi, Using Bayesian networks for cognitive control of multi-hop wireless networks, in: Military Communications Conference, 2010-Milcom 2010, IEEE, 2010, pp. 201–206.
[45] https://www.nsnam.org/
[46] I.A. Beinlich, H.J. Suermondt, R. Martin Chavez, G.F. Cooper, The ALARM Monitoring System: A Case Study With Two Probabilistic Inference Techniques For Belief Networks, Springer, Berlin Heidelberg, 1989.
[47] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, J. R. Stat. Soc. Ser. B (Methodological) (1988) 157–224.
[48] E. Herskovits, Computer-Based Probabilistic-Network Construction, Stanford University, 1991.
[49] Netica. Netica Bayesian network software from Norsys. http://www.norsys.com
[50] D. Cavalcanti, S. Das, J. Wang, K. Challapali, Cognitive radio based wireless sensor networks, in: 2008 Proceedings of 17th International Conference on Computer Communications and Networks, IEEE, 2008, pp. 1–6.
[51] Kok-LimAlvin Yau, P. Komisarczuk, P.D. Teal, Cognitive radio-based wireless sensor networks: conceptual design and open issues, in: 2009 IEEE 34th Conference on Local Computer Networks, IEEE, 2009, pp. 955–962.
[52] S. Kumar, D. Singhal, R.M. Garimella, Cognitive wireless sensor networks, Intell. Sens. Netw. (2012) 205.
[53] O.B. Akan, O.B. Karli, O. Ergul, Cognitive radio sensor networks, IEEE Netw. 23 (4) (2009) 34–40.
[54] G.P. Joshi, S.Y. Nam, S.W. Kim, Cognitive radio wireless sensor networks: applications, challenges and research trends, Sensors 13 (9) (2013) 11196–11228.
[55] S. Anusha, V. Mohanraj, Dynamic spectrum access in cognitive radio wireless sensor networks using different spectrum sensing techniques, Int. J. Appl. Eng. Res. 11 (6) (2016) 4044–4048.
[56] G.A. Shah, O.B. Akan, Cognitive adaptive medium access control in cognitive radio sensor networks, IEEE Trans. Veh. Technol. 64 (2) (2015) 757–767.
[57] I.F. Akyildiz, M.C. Vuran, O.B. Akan, A cross-layer protocol for wireless sensor networks, in: 2006 40th Annual Conference on Information Sciences and Systems, Ieee, 2006, pp. 1102–1107.
[58] S. Jagadeesan, V. Parthasarathy, Cross-layer design in wireless sensor networks, in: Advances in Computer Science, Engineering & Applications, Springer, Berlin Heidelberg, 2012, pp. 283–295.
[59] S. He, J. Chen, D.K. Yau, Y. Sun, Cross-layer optimization of correlated data gathering in wireless sensor networks, IEEE Trans. Mobile Comput. 11 (11) (2012) 1678–1691.
[60] F. Cuomo, A. Abbagnale, E. Cipollone, Cross-layer network formation for energy-efficient IEEE 802.15. 4/ZigBee wireless sensor networks, Ad Hoc Netw. 11 (2) (2013) 672–686.
[61] G.A. Shah, V.C. Gungor, O.B. Akan, A cross-layer QoS-aware communication framework in cognitive radio sensor networks for smart grid applications, IEEE Trans. Ind. Inform. 9 (3) (2013) 1477–1485.
[62] M.S. Hefeida, T. Canli, A. Khokhar, CL-MAC: a cross-layer mac protocol for heterogeneous wireless sensor networks, Ad Hoc Netw. 11 (1) (2013) 213–225.
[63] G. Han, Y. Dong, H. Guo, L. Shu, D. Wu, Cross-layer optimized routing in wireless sensor networks with duty cycle and energy harvesting, Wireless Commun. Mobile Comput. 15 (16) (2015) 1957–1981.
[64] B. Fu, Y. Xiao, H.J. Deng, H. Zeng, A survey of cross-layer designs in wireless networks, IEEE Commun. Surv. Tutorials 16 (1) (2014) 110–126.
[65] A. Coen-Porisini, S. Sicari, Improving data quality using a cross layer protocol in wireless sensor networks, Comput. Netw. 56 (17) (2012) 3655–3665.
[66] S. Yessad, L. Bouallouche-Medjkoune, D. Aïssani, A cross-layer routing protocol for balancing energy consumption in wireless sensor networks, Wireless Pers. Commun. 81 (3) (2015) 1303–1320.

S. Gheisari received the B.S. degree in Computer Engineering from the Islamic in 2006. She received the M.S. degree in Computer Network Engineering from Islamic Azad University of Qazvin in Iran in 2009. Currently, she has received her Ph.D. degree from department of computer in Science and Research University, Tehran, Iran. Currently she is an Assistant Professor in Computer Engineering Department, Pardis branch, Islamic Azad University, Pardis, Iran Her research interests include Computer networks, Cognitive networks, learning systems, Bayesian networks, and learning automata.

M. R. Meybodi received the B.S. and M.S. degrees in Economics from the Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, and Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include, channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.