Learning Automata Based Adaptive Petri Net and Its Application to Priority Assignment in Queuing Systems with Unknown Parameters

S. Mehdi Vahidipour, Mohammad Reza Meybodi and Mehdi Esnaashari

***Abstract*—in this paper an adaptive Petri net, capable of adaptation to the environmental changes, is introduced by the fusion of learning automata and Petri net. In this new model, called learning automata based Adaptive Petri Net (APN-LA), learning automata are used to resolve the conflicts among the transitions. In the proposed APN-LA model, transitions are portioned into several sets of conflicting transitions and each set of conflicting transitions are equipped with a learning automaton whose responsibility is to control the conflicts among transitions in the corresponding transition set. We also generalize the proposed APN-LA to ASPN-LA which is a fusion between LA and Stochastic Petri net (SPN). An application of the proposed ASPN-LA to priority assignment in queuing systems with unknown parameters is also presented.**

***Index Terms*—Adaptive Petri net, Conflict resolution, Learning Automata, Petri net.**

I. Introduction

PETRI nets (PNs) are graphical and mathematical modeling tool which have been applied to many different systems. They are used for describing and studying information processing systems with concurrent, asynchronous, distributed, parallel, nondeterministic and/or stochastic characteristics [1].

The evolution of a Petri net system can be described in terms of its initial marking and a number of firing rules [2]. At any given time during the evolution of a Petri net system, current marking of the system evolves to a new marking by applying the firing rules. Firing rules of an ordinary Petri net system, for any marking *M*, consists of three steps [2]: 1) determining the set of enabled transitions from available transitions, 2) selecting a transition from the set of enabled transitions for firing, and 3) generating a new marking by firing the selected transition.

Mechanisms which determine the set of enabled transitions from the available transitions in PN try to reduce the number of enabled transitions by introducing concepts such as inhibitor arc [3], priority[4]-[5], time [6], and color [7].

For selecting a transition for firing among the enabled transitions, some mechanisms are reported in the literature [3], [8]-[10]. Among these mechanisms, random selection [3] is the mostly used one. This is a good mechanism if there is no conflict among the enabled transitions. But, if two or more enabled transitions are in conflict, that is, firing one transition results in disabling the other(s), random selection will not be an appropriate method to resolve conflicts. Set of conflicts change when marking changes and hence random selection for selecting the transition to be fired is not appropriate when PN is used to model dynamics of real world problems [5]-[6]. To cope with this situation, one suitable approach is to add a mechanism in PN which can adaptively resolve conflicts among the enabled transitions.

One of the adaptive agents used in literature frequently is learning automaton (LA) which is a simple agent for making simple decisions [11]. In this paper, we propose an adaptive PN in which LA is used as a conflict resolution mechanism. The proposed adaptive Petri net, called learning automata based Adaptive Petri Net (APN-LA) is obtained from the fusion between PN and LA. In this model, at first, transitions are partitioned into several sets of conflicting transitions, each is called a cluster. Then, each cluster is equipped with a LA whose responsibility is to control the conflicts among transitions in the corresponding transition set. Each LA has a number of actions each of which corresponds to selection of one of the enabled transitions for firing. During the evolution of APN-LA, unlike standard PN, a cluster, instead of a transition, is selected for firing at each marking. A cluster can be selected for firing only if there exists at least one enabled transition within it. Then, LA associated with the cluster is activated for selecting one of the enabled transitions within the cluster. The selected transition is fired and a new marking is generated. Upon the next activation of this LA, a reinforcement signal will be generated considering the sequence of markings between the two activation of LA. Using the generated reinforcement signal, the internal structure of LA will be updated to reflect the marking's changes.

We also generalize the proposed APN-LA to ASPN-LA which is a fusion between LA and Stochastic Petri net (SPN). We

Manuscript received August 19, 2014; revised October 16, 2014 and January 09, 2015; accepted January 2015. This work was supported in part by a grant of the Iran Telecommunications Research Center (ITRC).

S. M. Vahidipour is with the Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran (e-mail: vahidipour@kashanu.ac.ir).

M. R. Meybodi is with the Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran (e-mail: mmeybodi@aut.ac.ir).

M. Esnaashari is with the Information Technology Department, ITRC, Tehran 1439955471, Iran (e-mail: esnaashari@itrc.ac.ir).

present an application of the proposed ASPN-LA to priority assignment in a queuing system under unknown characteristics. The queuing system consists of one server and two queues and the jobs are processed from the queues in such a way that the average waiting time of the system will be minimized. Finally, we introduce an ASPN-LA-[m]PP which models the priority assignment problem in a queuing system with m queues and one server, based on the proposed ASPN-LA.

The rest of the paper is organized as follows: Section II gives the basic definitions. The subject of related work is reviewed in Section III. In Section IV, the proposed adaptive Petri net is presented. In Section V, the proposed adaptive SPN is presented. Section VI gives an application of proposed ASPN-LA to design algorithms for priority assignment in an M/M/1 queuing system with unknown parameters.

II. Learning Automata and Petri nets

In this Section, we first briefly review learning automata and then briefly review Petri nets. A Learning automaton (LA) is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed action through repeated interaction with a random environment. The action is chosen at random based on a probability distribution kept over the action-set and at each instance the given action serves as the input to the random environment. The stochastic environment evaluates the taken action and then the automaton uses the environment response for selecting its next action (Fig. 1). The objective of a LA is to find the optimal action from the action set so that the average penalty received from the environment is minimized [12].

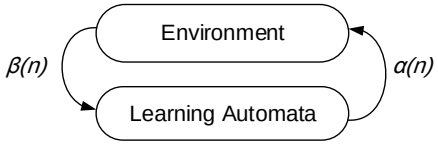

Fig. 1. Learning automata and environment.

Definition 1. Environment is shown by a triple $E = \{\alpha, \beta, c\}$, *where* $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$*represents the finite set of inputs,* $\beta = \{\beta_1, \beta_2, \dots, \beta_m\}$ *denotes the set of the values that can be taken by the reinforcement signal, and* $c = \{c_1, c_2, \dots, c_m\}$ *denotes the set of the penalty probabilities, where the element* c_i *is as associated with the given action*α_i. □

If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non-stationary environment. The environments depending on the nature of the reinforcement signal β can be classified into P-model, Q-model, and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of values in the interval [0, 1] can be taken by the reinforcement signal. Such an environment is referred to as a Q-model environment. In S-model environments, the reinforcement signal lies in the interval [a, b] [13].

LA can be classified into two main families [13]: fixed structure LA and variable structure LA. When the probability of the transition from one state to another state and the probability of any action in any state are fixed, LA is called fixed structure. In variable structure automata, automata's actions get updated in each repetition. In this type of automata, changing of actions probabilities is performed based on a learning algorithm.

Definition 2.Learning automaton (LA) with variable structure is represented by $\{\alpha, \beta, q, L\}$, *where* $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ *is the set of actions,* $\beta = \{\beta_1, \beta_2, \dots, \beta_m\}$ *is the set of inputs,* $q = \{q_1, q_2, \dots, q_r\}$ *is the action probability set and L is the learning algorithm.* □

The learning algorithm L is a recurrence relation which is used to modify the action probability vector. Let $\alpha(n)$ and $q(n)$ denote the action chosen at instant n and the action probability vector on which the chosen action is based, respectively. The recurrence equation shown by (1) and (2) is a linear learning algorithm by which the action probability vector q is updated. Let $\alpha(n)$ be the action chosen by the automaton at instant n

$$q_j(n+1) = \begin{cases} q_j(n) + a(n)\left[1 - q_j(n)\right], j = i \\ (1 - a(n))q_j(n) \qquad , \forall j \neq i \end{cases} \tag{1}$$

when the taken action is rewarded by the environment (i.e. $\beta(n) = 0$) and

$$q_j(n+1) = \begin{cases} q_j(n) + a(n)\left[1 - q_j(n)\right], j = i \\ (1 - a(n))q_j(n) \qquad , \forall j \neq i \end{cases} \tag{2}$$

when the taken action is penalized by the environment (i.e. $\beta(n) = 1$). r is the number of actions that can be chosen by the automaton, $a(n) \geq 0$ and $b(n) \geq 0$denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a(n) = b(n)$, the recurrence equations (1) and (2) are called the linear reward–penalty (L_{R-P}) algorithm, if $a(n) \gg b(n)$ the given equations are called the linear reward-ε penalty ($L_{R-\varepsilon P}$), andfinally if $b(n) = 0$ they are called the linear reward–Inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

At the time instant$n$, LA operates as follows: 1) LA randomly selects an action $\alpha(n)$ based on the action probability set $q(n)$, 2) performs $\alpha(n)$ on the environment and receives the environment's reinforcement signal $\beta(n)$, and 3) LA updates its action probability vector by the learning algorithm. Considering these operations, the time complexity of a LA with r actions is $O(r)$ because the time complexity of the first two steps is $O(1)$ and time complexity of the third step is $O(r)$.

LA have been found to be useful in systems where incomplete information about the environment where in the system operates, exists. LA are also proved to perform well in dynamic environments. It has been shown in [14] that the LA are capable of solving the distributed problems. Recently, several LA based approaches have been presented for improving the performance of many applications [15]-[17].

In [18]-[20] two kinds of LA are presented in which the number of actions available for selection may be changed over time. For example in the learning automaton proposed in [18] when the selection probability of any action becomes zero, the action is removed from the available set of actions from that time on. In the LA introduced in [19]-[20], the set of available actions can be varied at any instance of time. This learning automaton, called LA with variable set of actions, can be formally defined as below.

Definition 3. LA with variable set of actions is a LA that has a finite set of r actions $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$. $A = \{A_1, A_2, \dots, A_{2^r-1}\}$ denotes the set of action subsets and $A(n)$ is the subset of all the available actions which can be chosen by LA, at instant n. $\hat{q}_i(n)$ denotes the probability of choosing action α_i, conditioned on the event that the action subset $A(n)$ has already been selected and $\alpha_i \in A(n)$ too. The scaled probability $\hat{q}_i(n)$ is defined as

$$\hat{q}_i(n) = prob[\alpha(n) = \alpha_i | A(n), a_i \in A(n)] = q_i(n)/K(n) \quad (3)$$

where $K(n) = \sum_{\alpha_i \epsilon A(n)} q_i(n)$ is the sum of the probabilities of actions in subset $A(n)$, and $q_i(n) = prob[\alpha(n) = \alpha_i]$□

The procedure of choosing an action and updating the action probabilities in a variable action set LA can be described as follows [19]: Let $A(n)$ be the available action set selected at instant n. Before choosing an action, the probabilities of all the actions in the available set are scaled as defined in equation (3). The automaton then randomly selects one of its possible actions according to the scaled action probability vector $\hat{q}(n)$. Depending on the response received from the environment, LA updates its scaled action probability vector: $\hat{q}(n+1) = L(\hat{q}(n), \alpha(n), \beta(n))$. Note that the probability of the available actions is only updated. Finally, the probability vector of the actions of the chosen subset is rescaled as defined in equation (4) for all $\alpha_i \in A(n)$.

$$q_i(n+1) = \hat{q}_i(n+1) * K(n) \text{ if } a_i \in A(n) \quad (4)$$

Definition 4. Petri net (PN) is a triple $\{P, T, W\}$, where P is a non-empty finite set of places, T is a non-empty finite set of transitions, and $W: ((P \times T) \cup (T \times P)) \to \mathbb{N}$ defines the interconnection of P and T sets.□

For each element x, either a place or a transition, its preset is defined as $\bullet x = \{y \in P \cup T | W(y,x) > 0\}$ and its post-set is defined as $x\bullet = \{y \in P \cup T | W(x,y) > 0\}$. A Marking M is $|P|$-vector and $M(i)$ is the non-negative number of tokens in place P_i. A transition t is defined to be enabled in marking M (denoted $[M, t>$) if, for every place $p_i \in \bullet t$, $M(i)$ is equal or greater than $W(p,t)$. Transition t can fire if $[M, t>$; the firing operation is denoted as $M[t > M'$, meaning that $[M, t>$ and that M' is the next marking in PN evolution. A PN along with an initial marking M_0 creates PN system.

The set of markings reached from M_0 is called a reach-ability set and represented by a reach-ability graph [1]-[2], [6].

Definition 5. Stochastic Petri net (SPN) is a PN and is defined by a 6-tuple $\{P, T, W, R, \omega, M_0\}$, where

- *P, T, W, M_0 are defined as in Definition 4.*
- *$\forall t \epsilon T, R_t \in \mathbb{R}^+ \cup \{\infty\}$ is the rate of exponential distribution for the firing time of transition t. If $R_t = \infty$, the firing time of t is zero; such a transition is called immediate. On the other hand, a transition t with $R_t < \infty$ is called timed transition. A marking M is said to be vanishing if there is an enabled immediate transition in this marking; M is said to be tangible otherwise [21].*
- *$\forall t \epsilon T, \omega_t \in \mathbb{R}^+$ describes the weight assigned to the firing of the enabled transition t, whenever its rate R_t evaluates to ∞. The firing probability of transition t enabled in a vanishing marking M is computed as $\frac{\omega_t}{\sum_{[M,t'>} \omega_{t'}}$.* □

It should be noted that the above definition of SPN coincides with the definition of GSPN given in [6].

III. Related work

Control for concurrent processes may be needed with respect to synchronization and conflict resolution. For PN, control can be thought as the mechanism which specifies the order by which transitions are fired so that probable conflicts among transitions can be resolved. The mostly used controlling mechanism in literature is the random selection[3], which as mentioned before, is not appropriate in conflicting situations. Queue regimes [8] are some other mechanisms used for controlling the order of firing in PN in which, enabled transitions are first put in a queue and then a number of different approaches are proposed to be used for selecting a transition from this queue for firing. These mechanisms are again not appropriate for conflicting situations.

In priority net, a priority level is assigned to each transition. In any markings, an enabled transition can be selected for firing only if no enabled transition with higher level of priority exists [22]-[23]. If priority level of a transition does no change in different markings, then the priority net is called static [23]. Otherwise, it is called dynamic[24]-[25]. In a dynamic priority net, the priority level of a transition in one marking is a fix value and this value can be modified in different markings. This is again not a suitable adaptive controlling mechanism for conflicting situations, especially in dynamic environments.

A model of PN controlled by finite automata is proposed in [26] in which an automaton controls the firing mechanism of PN. This automaton interacts with PN in two steps; in the first step, the set of all enabled transitions in the current marking of PN form the action set of the automaton. In the second step, the firing transition in PN is determined using the automaton's transition function whenever the next state of automaton is also determined by this function. In [26], authors used a number of finite automata, instead of only one automaton, for controlling the firing mechanism of PN. In their proposed method, all transitions in PN are partitioned into a number of disjoint sets. Each automaton is then responsible for controlling the firing mechanism among one of these disjoint sets. This mechanism may be able to handle conflicts among transitions, but due to the deterministic nature of finite automaton, it cannot adapt itself to the environmental changes.

Several authors have used a particular PN, called controlled Petri nets, within the framework of supervisory control [27]-[28] where PN is used as a discrete event model [29]-[30]. Controlled Petri nets (CtlPNs) are a class of PN with a new

kind of places called control places. A control place allows an external controller to influence the evolution of PN. In most cases, the external controller must control the evolution of CtlPN so that the system always remains in a specified set of allowed states, or, equivalently, a set of forbidden markings are never reached by the CtlPN. Such a control policy can be designed by some general approaches [31] such as linear integer programming [32] and path-based approaches [33].

One drawback of the aforementioned mechanisms is the lack of learning capacity which prevents these mechanisms from adapting themselves to the changes occurred in environment. Of course, there are approaches in literature which try to make PN adaptive through fusing intelligent techniques [34] such as neural networks and fuzzy logic systems with PN [35]-[37]. But none of these methods is an adaptive controlling mechanism which can be used in conflicting situations. For example in fusion between PN and neural networks, proposed in [38], a feed-forward neural Petri net (NPN) is constructed in which the basic concepts of PN (place, transition and arc) are utilized. Places in NPN are decomposed into three layers: input, hidden and output. In order to construct connections between places in hidden and output layers, thresh-holding transitions and arcs with trainable weights are introduced. A thresh-holding transition is enabled when the summation of its weighted inputs is equal to or greater than its predefined threshold. Using a set of input-output training examples, the weights of arcs are trained according to the back propagation rules. Another example of fusion between PN and neural networks is recently reported in[39]. In this paper, a special transition, called Adaption transition (A-transition), is introduced which is associated with a multilayer neural network. The inputs of the neural network are the markings of its input places whereas the outputs are the markings of its output places. The inputs of all A-transitions create an environment for the system and, thus, when the environmental changes happen, A-transitions can adapt themselves with the changes. Based on the markings of the output places of a fired A-transition, a sub-graph of the reachability graph will be traversed. Different types of fusion between PNs and neural networks are also proposed in literatures[40]-[41].

As another example, fusion between PN and Fuzzy logic systems is reported in [42] where PN is used to represent fuzzy rule-based systems. In this representation, places represent propositions and tokens represent states of the propositions. A value between 0 and 1 is assigned to each token representing the truth degree of proposition state. Firing a transition expresses a rule reasoning process [43]-[44]. Combination of LA with Color fuzzy Petri net (CFPN) is recently proposed in [45]. In this combination, LA is only used to adaptively adjust fuzzy functions defined on the input parameters. Therefore, it cannot be considered as a controlling mechanism for PN.

IV. ADAPTIVE PN BASED ON LA (APN-LA)

An adaptive Petri net based on learning automata (APN-LA) is a PN in which conflicts among transitions are resolved using learning automata. LA is a simple model for adaptive decision making in unknown random environments [11]. In addition, this model is proved to perform well in dynamic environments [13]. Thus, we believe that LA can be a good candidate for adopting an adaptive controlling mechanism in PN. In this Section, first an APN-LA will be formally defined, and then, to describe the evolution of APN-LA, its firing rules will be presented.

Before giving the formal definition of APN-LA, we first propose a number of useful definitions.

Definition 6. Potential Conflict is a set of transitions $\{t_1, t_2, \cdots, t_n\}$ *iff every* $t_k \in \{t_1, t_2, \cdots, t_n\}$ *shares at least one input place with one* $t_j \in \{t_1, t_2, \cdots, t_n\}/\{t_k\}$. *The potential conflict only depends on the structure of PN graph.* □

Definition 7. Maximal Potential Conflict is a set of transitions $\rho = \{t_1, t_2, \cdots, t_n\}$ *iff following conditions are held:*

- $\{t_1, t_2, \cdots, t_n\}$ *are in potential conflicts,*
- *For every* $t \in T \backslash \rho$, $\{t\} \cup \rho$*aren'tin potential conflict.* □

We refer to a maximal potential conflict set as a cluster hereafter.

Definition 8. An updating transition,$t^u \in T^u$, *is an ordinary immediate transition, except that when it fires, the action probability vector of a LA, fused with PN, is updated.* □

A. Formal Definition

An APN-LA can be formally defined as below:

Definition 9. APN-LA is a 5-tuples $(\hat{P}, \hat{T}, \widehat{W}, S, L)$, *where* $\hat{P}$ *is a finite set of places,* $\hat{T} = T \cup T^U$ *is a finite set of ordinary transitions and updating transitions* $T^U = \{t_1^u, \ldots . t_n^u\}$, $\widehat{W}: ((\hat{P} \times \hat{T}) \cup (\hat{T} \times \hat{P})) \to \mathbb{N}$ *defines the interconnection of* $\hat{P}$ *and*$\hat{T}$, $L = \{LA_1, \ldots . LA_n\}$ *is a set of learning automata with varying number of actions, and* $S = \{s_0, s_1, \ldots . s_n\}$ *denotes a set of clusters, each consists of a set of transitions:*

- $s_i, i = 1, \ldots, n$, *are sets of clusters, each is a maximal potential conflict set. Each cluster* s_i *is equipped with a learning automaton* LA_i. *Number of actions of* LA_i *is equal to the number of transitions in* s_i*; each action corresponds to one transition.*
- s_0*is the set of remaining transitions in* $\hat{T}$.□

Definition 10. APN-LA system is a triple$(\widehat{N}, M_0, \hat{F})$, *where*$\widehat{N}$ *is an APN-LA,*M_0 *is the initial marking,*$\hat{F} = \{f_1, \cdots, f_n\}$ *is the set of reinforcement signal generator functions.* $f_i: M \to \beta_i$*is the reinforcement signal generator function related to* LA_i. *Sequence of markings in APN-LA is the input to* f_i *and reinforcement signal* β_i *is its output. Upon the generation of*β_i, LA_i *updates its action probability vector using the learning algorithm given in equation (3) or (4).* □

B. Evolution of APN-LA System

The evolution of a Petri net system can be described in terms of its initial marking and a number of firing rules [2]. At any given time during the evolution of a Petri net system, current marking of the system evolves to a new marking by applying the firing rules. Firing rules of an ordinary Petri net system, for any marking *M*, are as follows[2]:

1. A transition $t \in T$ is said to be enabled if each input place p of t is marked with at least $W(p,t)$ tokens.
2. An enabled transition may or may not fire (Only one enabled transition can fire in each marking).
3. A firing of an enabled transition t removes $W(p,t)$ tokens from each input place $p$of t, and adds $W(t,p)$ tokens to each output place p of t.

Firing rules of an APN-LA system differs from that of an ordinary PN due to the fusion with LA. Therefore, in this Section, we will introduce the firing rules for APN-LA. First we give a number of definitions which will be used later for defining the firing rules of APN-LA.

Definition 11. Enabled cluster s_i is enabled in marking M when at least one transition in s_i is enabled.□

Definition 12. Fired cluster is an enabled cluster which is selected as the fired cluster, from which a transition will be fired in marking M.□

Definition 13. $LA_i$is activated in marking M iff s_i is selected as fired cluster in marking$M$.□

Definition 14. Effective Conflict: a subset E_M^s of a cluster $s$is said to be in effective conflict in marking M iff these conditions are held: $\forall t \in E_M^s$is enabled in M and$\{\forall t' \in s | t' \notin E_M^s, [M,t' \succ \}$. □

Considering the above definitions, the firing rules of an APN-LA in any marking M can be described as follows:

1. A transition $t \in \hat{T} = T \cup T^U$ is said to be enabled if each input place p of t is marked with at least $\widehat{W}(p,t)$ tokens.
2. A cluster $s_i$is said to be enabled if at least one transition in s_i is enabled.
3. If s_0 is enabled, then s_0 is selected as fired with probability$\frac{|t' \in s_0, [M,t'>|}{|t \in \hat{T}, [M,t>|}$, where $|\quad|$ stands for norm operator. Otherwise, an enabled cluster $s_i, i = 1, \dots, n$ is selected as fired with probability $\frac{|E_M^{s_i}|}{|t \in \hat{T}, [M,t>|}$ (Only one enabled cluster can be selected as the fired cluster in each marking).
4. Only one enabled transition $t \in \hat{T}$ from the fired cluster can fire in each marking M.
5. If s_0 is the fired cluster, then an enabled transition t is randomly selected from $s_0$for firing. Otherwise, LA associated to the fired cluster is activated. The available action set of the activated LA consists of the actions corresponding to the enabled transitions ($E_M^{s_i}$) in the fired cluster. The activated LA selects one action from its available action set randomly according to its action probability vector and then the corresponding transition is selected for firing.
6. Firing of a transition $t \in \hat{T}$ removes $\widehat{W}(p,t)$ tokens from each input place p of t, and adds $\widehat{W}(t,p)$ tokens to each output place p of t.
7. The reinforcement signal generator function $f_i \in \hat{F}$ is executed when the updating transition $t_i^u \in T^u$fires.
8. The reinforcement signal for the activated LA will be generated at the next activation of that LA.
9. Using the generated reinforcement signal, the internal structure of LA will be updated to reflect the marking's changes.

It is worth mentioning here that two or more transitions may be enabled at the same time without sharing any input place. For example, consider two transitions t_2 and t_3 in Fig. 2. Although these transitions are in actual conflict in marking $\{p_2+p_3\}$, the potential conflict condition defined in Definition 6 does not hold for these transitions; our proposed APN-LA does not resolves this actual conflict. As a consequence, the proposed APN-LA, as defined above is only applicable for conflict resolution among the effective conflicts.

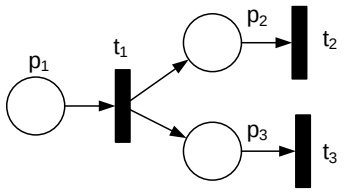

Fig. 2. Two transitions t_2 and t_3 can be enabled at the same time without sharing any input place.

V. Generalization of APN-LA

In this Section, we generalize the proposed APN-LA to ASPN-LA which is a fusion between LA and Stochastic Petri net (SPN). The most important difference between PN and SPN is the addition of timed transitions. A timed transition can be assumed to possess a countdown timer, which is started whenever the transition becomes enabled. This timer is set to a value that is sampled from the negative exponential probability density function associated with the transition. The transition is fired whenever its timer reaches zero. Thus, no special mechanism is necessary for the resolution of timed conflicts: the temporal information provides a metric that allows the conflict resolution [6]. As such, for generalizing APN-LA to ASPN-LA, it is sufficient to put all timed transitions of SPN into a single cluster, referred to ass_{-1}, in which the temporal information is used to select one enabled timed transition for firing. Formally, an ASPN-LA can be defined as follows:

Definition 15. ASPN-LA is a 6-tuples$\left(\hat{P}, \hat{T}, \widehat{W}, \hat{S}, L, R, \omega^0\right)$, *where*

- $\hat{P}, \hat{T}, \widehat{W}, L$ *are defined as in Definition 9 and R is defined as in Definition 5.*
- $\hat{S} = \{s_{-1}, s_0, s_1, \dots . s_n\}$ *denotes a set of clusters, each consists of a set of transitions:*
- s_{-1} *consists of all timed transitions in ASPN-LA.*
- s_0 *and* $s_i, i = 1, . , n$ *are defined as in Definition 9.*
- $\forall t \epsilon s_0, \omega_t^0 \in \mathbb{R}^+$ *describes the weight assigned to the firing of enabled transition* t *in clusters*s_0. □

Note that the definition of ω_t in SPN is changed into ω_t^0 in ASPN-LA. In other words, in ASPN-LA, weights are only assigned to the immediate transitions in the clusters_0. This is due to the fact that in ASPN-LA, conflict resolutions among immediate transitions in $s_i, i = 1, \dots, n$ are performed by LA, and hence, no weight is required for these set of transitions.

Firing rules of an ASPN-LA system differs from that of an

APN-LA due to the addition of the cluster s_{-1}. Therefore, following rules must be added to the firing rules of APN-LA:

1. Cluster s_{-1} is said to be enabled if at least one timed transition in s_{-1} is enabled and no other cluster $s_i, i \geq 0$ is enabled.
2. Cluster s_{-1} is selected as fired cluster if it is enabled.
3. If s_{-1} is the fired cluster, the temporal information associated with the enabled transitions within s_{-1} is used for selecting a transition for firing.

In addition, if s_0 is selected as the fired cluster, then instead of selecting a transition at random for firing, an enabled transition t is selected for firing with probability $\frac{\omega^0{}_t}{\sum_{[M,t'>} \omega^0{}_{t'}}$.

It is worth mentioning here that in an SPN, if several immediate transitions are enabled, a metric is necessary to identify which transition will produce the marking modification. Of course, this is required only in those cases in which a conflict must be resolved; if the enabled transitions are concurrent, they can be fired in any order [6]. Weights, assigned to immediate transitions, are used as the metric for determining which immediate transition will actually fire in conflicting situations. A practical approach for assigning weights to conflicting transitions is given in [6]. The main difference between the proposed ASPN-LA and GSPN is that, the conflict resolution mechanism in ASPN-LA is adaptive to the environmental changes, whereas weights and priorities, assigned to immediate transitions in GSPN, are constant.

VI. An application of ASPN-LA

In this Section, we use the proposed ASPN-LA to solve the priority assignment (PA) problem in a queuing system with one server and some classes of jobs. First, we concentrate to construct ASPN-LA model for a queuing system with two classes of jobs and then, we study the steady state behavior of this model. Finally, we introduce an ASPN-LA-[m]PP which models PA problem in a queuing system with m queues and one server, based on the proposed ASPN-LA.

A. PA Problem in a Queuing System with Two Queues

In a queuing system with one server and two queues, there are two classes of jobs. Jobs from class i arrive into queue Q_i at constant Poisson rate $\lambda_i, i = 1, 2$. The service times of these classes are independent and identically distributed with exponential distribution having mean $\mu_i^{-1}, i = 1, 2$. Fig. 3 illustrates the abstract model of the queuing system with two classes of jobs.

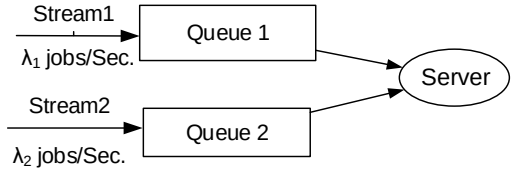

Fig. 3. Queuing system with one server and two queues.

When the server becomes idle, it has to select a new job from either of the two queues. In other words, when both queues have jobs, waiting for service, server must prioritize one class of jobs to the other class. This prioritization affects the total waiting time of the system [46], and is referred to as priority assignment (PA) problem. That is, PA problem in a queuing system with two queues is as follows: how to select jobs from the two queues so that the total waiting time of the system is minimized.

In a queuing system with two queues Q_1 and Q_2, if $\mu_1 > \mu_2$, then it means that the average service time of the jobs arriving into Q_1 is lower than that of Q_2. In this queuing system, the lower total waiting time of the system is obtained when the server selects jobs from Q_1 with higher priority [46]-[47]. If μ_1 and μ_2 are known to the server, then PA problem can be solved by a simple mechanism such as strict priority [48]; the server has to always select jobs from Q_1 with higher priority. But, if μ_1 and μ_2 are unknown, then such a simple mechanism is not adequate. One way to solve PA problem in this case, is by adopting a probabilistic priority (PP) mechanism [49]. In this mechanism, the priority of each class is defined by a probability with which its corresponding queue is selected. Here, the lower total waiting time of the system is obtained when the server selects jobs from Q_1 with higher probability (probabilistic priority) rather than higher priority.

One simple PP mechanism to solve PA problem can be described using the SPN, referred to as SPN-PP (Fig. 4), in which server selects jobs from either of the queues with equal probability of 0.5. In this SPN-PP, p_1 represents the server and p_2 and p_4 represent Q_1 and Q_2 respectively. If the server is idle ($M(p_1) = 1$) and both queues have jobs, ($M(p_2) > 1$ and $M(p_4) > 1$), then one of the jobs is selected for execution by pure chance. In other words, in this SPN-PP, no priority assignment scheme is used to decrease the total waiting time of the system. A better PP mechanism can be obtained by adopting APN-LA instead of SPN for solving PA problem. This model will be described in the following Section.

B. Modeling PA Problem with ASPN-LA

First, an ASPN-LA is constructed based on SPN of Fig. 4. The set $\{t_5, t_6\}$ is a maximal potential conflict, and hence, forms a cluster s_1. The transitions t_5 and t_6 are in effective conflict in marking $M = p_1 + p_2 + p_4$. Hence, we assign LA_1 with two actions to the cluster s_1; each action corresponds to select one transition for firing. After construction of ASPN-LA from SPN model, the updating transition t_1^u, input place p_6 and output place p_7 are inserted into the model. The newly output place p_7 is connected to updating transition t_1^u with an inhibitor arc. The place p_7 is added to the preset of t_5 and t_6. Since place p_1 is the shared input place of transitions t_5 and t_6, we add preset of this place to preset of p_6. A token is appeared in p_7 because there exists a token in place p_1, the shared input place of the transitions t_5 and t_6. Finally, transitions t_1, t_2, t_3, and t_4 create the cluster s_{-1} and t_1^u creates s_0. The newly generated ASPN-LA is called ASPN-LA-PP and is shown in Fig. 5.

To obtain ASPN-LA-PP system, the reinforcement signal generator function set $\hat{F} = \{f_1\}$ is needed. f_1 is executed upon the firing of t_1^u and generates the reinforcement signal β_1 for LA_1. To specify how f_1 generates β_1, we first note that when t_1^u

fires for the n^{th} time, following parameters are known to the system:

1. The queue from which the last job (job *n*) was selected for execution,
2. The execution time of job *n*, referred to as $\delta(n)$,
3. Number of jobs from $Q_i$which have been given service so far, referred to as $k_i(\sum_i k_{\mathrm{i}} = n)$,
4. Total service time for the jobs in Q_i which have been given service so far, referred to as $\Delta_i(n)$,
5. Average service time for the jobs in Q_i which have been given service so far, which can be calculated as $\Gamma_i(n) = \frac{1}{k_i} \times \Delta_i(n)$.

Considering the above parameters,f_1 can be described using equation (5) given below:

$$\begin{cases} \beta_1 = 1;\ \delta(n) \geq \frac{1}{2} \times [\Gamma_1(n) + \Gamma_2(n)] \\ \beta_1 = 0;\ \delta(n) < \frac{1}{2} \times [\Gamma_1(n) + \Gamma_2(n)] \end{cases} \quad (5)$$

where$\beta_1 = 1$ is the penalty signal and $\beta_1 = 0$ is considered as a reward. Using β_1, generated according to equation (5), LA_1updates its available action probability vector, $\hat{q}(n)$, to the $L_{\mathrm{R-I}}$ learning algorithm described in equation (1). Then the probability vector of the actions of the chosen subset is rescaled according to equation (4).

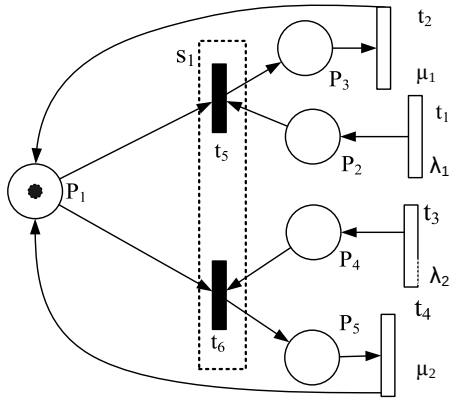

Fig. 4. A model of PP mechanism to solve PA problem using SPN.

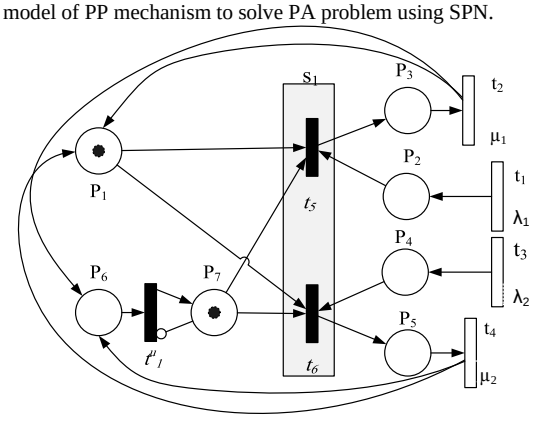

Fig. 5. ASPN-LA-PP models the PA problem in a queuing system with two queues and one server

C. The analysis of ASPN-LA-PP system

Since the fusion of LA with PN does not change the reachability graph of the underlying PN, any method which can be used for analyzing the underlying PN is also applicable for analyzing the fused version of that PN with LA. This is why we have been able to analyze ASPN-LA using its underlying CTMC, which is used for analyzing Stochastic Petri net (SPN). Thus, in what follows, we first describe how a CTMC can be derived from the proposed ASPN-LA-PP system and then, use this CTMC to study the steady-state behavior of the system.

To analyze the learning ability of ASPN-LA-PP, we will show that LA_1associated with cluster s_1 gradually learns to assign higher probabilities to the first class of jobs in comparison to the second class, if $\mu_1 > \mu_2$. To this end, we will partition the states of the yielded CTMC into three groups: 1) a job is selected from Q_1while another job is waiting in Q_2, 2) a job is selected from Q_2while another job is waiting in Q_1, and 3) remaining states. We can say that LA_1learns when the summation of the steady-state probabilities of states in the first group is higher than that of the second group. We will show that this can happen if the learning algorithm L_{R-I} is used for updating the action probability vector $q(n)$ of LA_1 in ASPN-LA-PP system.

1) Deriving CTMC from ASPN-LA-PP

ASPN-LA-PP is a SPN which contains only immediate and timed transitions, and hence an extended reachability graph (ERG) can be generated from it [50]. Each node in this ERG is a reachable marking of ASPN-LA-PP, and each directed arc from M_i to $M_j$represents the probability or rate of reaching to $M_j$from M_i. ERG can be transformed into a reduced reachability graph (RG) by eliminating the vanishing marking and the corresponding transitions [50]. Finally a CTMC will result from RG [6].

Let L_i denotes the number of tokens in place $p_i$and $M_j = (L_1, \cdots, L_7)$ denotes the j^{th} marking of the ASPN-LA-PP. The ERG, obtained from the initial marking $M_0 = (1,1,0,1,0,1,0)$, is shown in Fig. 6(a). To reduce the number of markings in this ERG, we assume that places p_2 and p_4 are one-bounded [1]. This ERG contains seven vanishing markings $\{M_0, M_2, M_6 - M_8, M_{10}, M_{12}\}$ represented by rectangles and thirteen tangible markings $\{M_1, M_3 - M_5, M_9, M_{11}, M_{13} - M_{15}, M_{16} - M_{19}\}$represented by ovals.

In Fig. 6(b), the finite and irreducible CTMC corresponding to Fig. 5 is shown. In this CTMC each state j corresponds to the tangible marking M_j of the ERG given in Fig. 6(a).

2) Steady-State analysis of ASPN-LA-PP

From Fig. 6(a) we can see that in some tangible markings, a job is selected from Q_1while another job is waiting in Q_2, i.e. marking M_1 and M_4 which are shown in Fig. 6(a) by bold ovals. In other words, in these markings, the ASPN-LA-PP system assigns higher probability to the first class than to the second class. The set of markings in which higher probabilistic priority is assigned to the first class is $S^{(1)} = \{s_1, s_4\}$which are shown in Fig. 6(b) by bold circles. On the other hand, in the set of markings $S^{(2)} = \{s_3, s_5\}$, ASPN-LA-PP system gives probabilistic priority to the second class.

With the CTMC in Fig. 6(b), the steady-state probability vector π can be derived using equation $\pi\mathbb{Q} = 0$ where $\mathbb{Q}$ is the infinitesimal generator matrix [50]. Letting π_i denotes the steady-state probability of the CTMC being in state i and letting $P^{(i)}$denotes the steady-state probability of marking

set $S^{(i)}, i = 1,2$. We conclude that$P^{(1)} = \pi_1 + \pi_4$and $P^{(2)} = \pi_3 + \pi_5$.

Under assumption $\mu_1 > \mu_2$, we analyze our proposed ASPN-LA-PP system by comparing the values of $P^{(1)}$ and $P^{(2)}$ when the system reaches to the steady state; if $P^{(1)} > P^{(2)}$ then PP mechanism assigns the higher probability to first class of jobs.

In this Section, we design two theorems to prove that ASPN-LA-PP reaches to the steady states in which $P^{(1)} > P^{(2)}$.In Theorem 1 by using the steady-state probabilities, we calculate a general lower bound value based on μ_1 and μ_2 such that if the value of $q_1(n)$ passes this value, then ASPN-LA-PP gives higher probabilistic priority to first queue. In Theorem 2, we show that if LA_1 uses an L_{R-I} algorithm to update its action probability vector, and then $P^{(1)} > P^{(2)}$holds when n goes to infinity. In order to follow CTMC analysis by ordinary methods, the stochastic information attached to the arcs should be fixed values. This is why we have assumed that fixed values q_1^* and q_2^* instead of $q_1(n)$ and $q_1(n)$ respectively.

Theorem 1: In order for ASPN-LA-PP to assign higher probabilistic priority to the first class of jobs, the inequality$q_1^* > \frac{\mu_1}{\mu_1+\mu_2}$ must be held.

Proof: Appendix A.

Corollary 1: under assumption$\mu_1 > \mu_2$, the lower total waiting time in ASPN-LA-PP is obtained when the assigned probabilistic priority to first queue passes $\frac{\mu_1}{\mu_1+\mu_2}$ value. In other words, the values of $q_1(n)$ must be adjusted to a value upper than $\frac{\mu_1}{\mu_1+\mu_2}$by LA_1.

Theorem 2: under assumption $\mu_1 > \mu_2$in ASPN-LA-PP system, if algorithm L_{R-I} is used to update action probability vector $q(n)$of LA_1, then the relation $P^{(1)} > P^{(2)}$holds when n goes to infinity.

Proof: Appendix B.

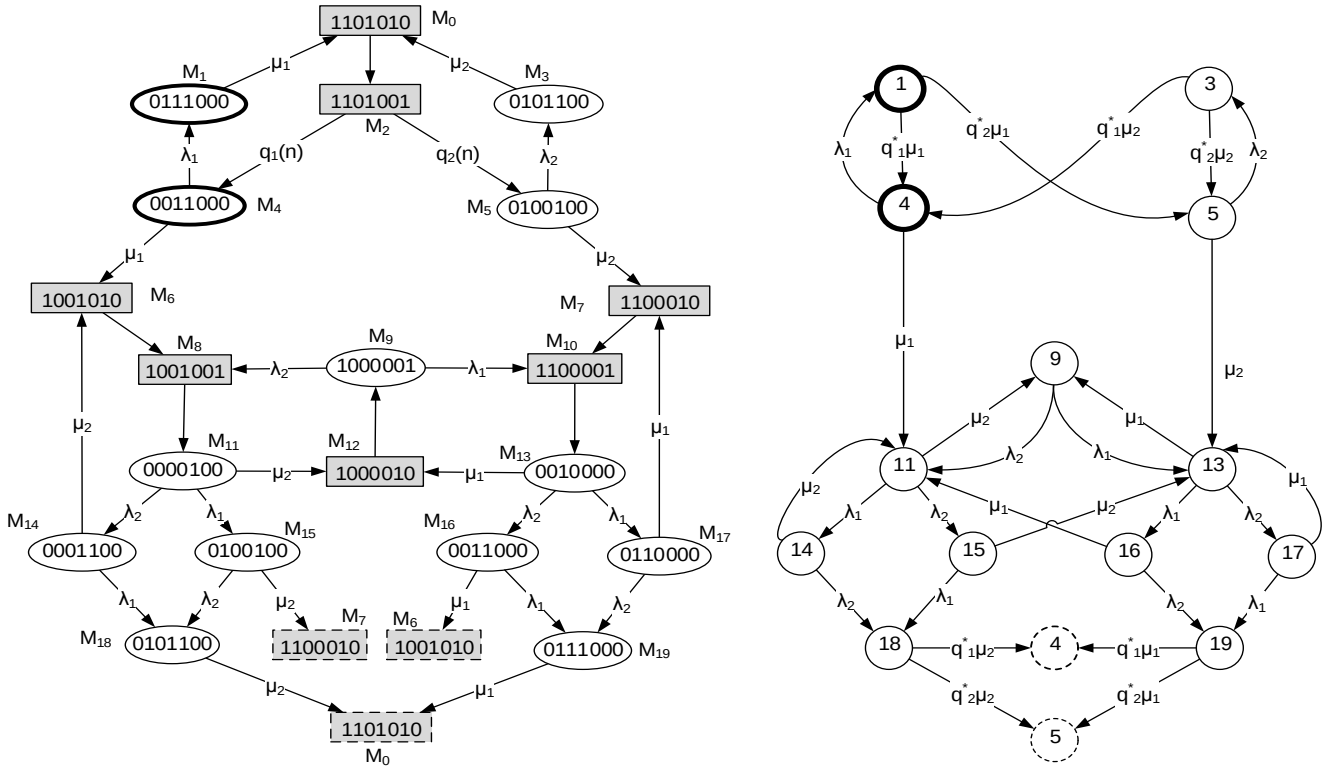

Fig. 6. (a). The extended reachability graph of ASPN-LA-PP in Fig. 5, (b). CTMC for ASPN-LA-PP in Fig. 5. In the highlighted states (markings), ASPN-LA-PP system assigns higher probability to the first class than to the second class. For simplicity, some states (Markings) has been duplicated (indicated by dash-line).

D. Simulation results

In this Section, we conduct a set of computer simulations to study the behavior of ASPN-LA-PP in comparison to that of SPN-PP in terms of the average waiting time of the queuing system. To have stable queues, in all simulations, we consider $\rho_i = \frac{\lambda_i}{\mu_i}, i = 1,2$ being less than 1. Without loss of generality, we assume$\mu_1 > \mu_2$. Let λ_1,λ_2,μ_1, and μ_2 to take one of the following values: .2, .4, .6, .8, or 1. All reported results are averaged over 100 independent runs of simulation. Each run of simulation consists of servicing 15,000 jobs.

1) Experiment 1

In this experiment, we study the average waiting times of different queuing systems resulted from ASPN-LA-PP and SPN-PP. The results of this experiment are reported in TABLE I. Each row in this table represents a queuing system with specific values of λ_1, λ_2, μ_1, and μ_2. The average waiting times of each queuing system resulted from SPN-PP and ASPN-LA-PP are given in the last two columns of the corresponding row of that queuing system. As it can be seen from this Table 2, when LA takes part in priority assignment, that is ASPN-LA-PP, the average waiting time of the system is

TABLE I
THE AVERAGE WAITING TIMES OF DIFFERENT QUEUING SYSTEMS RESULTED FROM ASPN-LA-PP AND SPN-PP

Set of parameters				Mean time of service (Sec.)	
λ_1	μ_1	λ_1	μ_1	SPN-PP	ASPN-LA-PP
0.8	*1*	*0.6*	*0.8*	*24.59*	***10.19***
0.8	*1*	*0.4*	*0.8*	*14.64*	***7.53***
0.8	*1*	*0.2*	*0.8*	*3.56*	***2.74***
0.6	*1*	*0.6*	*0.8*	*18.6*	***11.53***
0.6	*1*	*0.4*	*0.8*	*6.04*	***5.11***
0.6	*1*	*0.2*	*0.8*	*0.12*	***0.11***
0.4	*1*	*0.6*	*0.8*	*8.3*	***8.28***
0.4	*1*	*0.4*	*0.8*	*0.19*	***0.18***
0.4	*1*	*0.2*	*0.8*	*0.05*	***0.04***
0.2	*1*	*0.6*	*0.8*	*0.065*	***0.064***
0.2	*1*	*0.4*	*0.8*	*0.034*	***0.033***
0.2	*1*	*0.2*	*0.8*	*26.05*	***11.3***
0.6	*0.8*	*0.4*	*0.6*	*6.72*	***4.68***
0.6	*0.8*	*0.2*	*0.6*	*12.8*	***10.51***
0.4	*0.8*	*0.4*	*0.6*	*0.15*	***0.13***
0.4	*0.8*	*0.2*	*0.6*	*0.25*	***0.24***
0.2	*0.8*	*0.4*	*0.6*	*19.64*	***9.63***
0.4	*0.6*	*0.2*	*0.4*	*25.38*	***7***
0.8	*1*	*0.4*	*0.6*	*9.04*	***3.93***
0.8	*1*	*0.2*	*0.6*	*25.03*	***4.8***
0.8	*1*	*0.2*	*0.4*	*18.7*	***8.68***
0.6	*1*	*0.4*	*0.6*	*0.3*	***0.26***
0.6	*1*	*0.2*	*0.6*	*10.45*	***3.68***
0.6	*1*	*0.2*	*0.4*	*5.3*	***4.16***
0.4	*1*	*0.4*	*0.6*	*0.08*	***0.07***

lower than in SPN-PP where no learner presents in priority assignment process.

2) Experiment 2

This experiment is conducted to study the priority assignment behavior of the ASPN-LA-PP. To this end, we define n^* as the number of jobs, after which probabilistic priority of Q_1 is higher than that of Q_2, that is, $P^{(1)} > P^{(2)}$. We let $\mu_1 = 1.0$ and $\mu_2 = .8$ and change the values of ρ_1 and ρ_2 in the range [.2, .8]. Fig. 7provides the results of this experiment. In this figure, region A, which is the region below the dotted-line, contains the results of the experiment when $\rho_1 > \rho_2$ and region B (above the dotted-line) contains the results when $\rho_1 < \rho_2$. From this figure, one may conclude the following points:

- n^*assumes lower values in region A in comparison to region B. This is due to the fact that when$\rho_1 > \rho_2$, server gives services to jobs in Q_1 more frequently than the time when $\rho_1 < \rho_2$, and hence it can learn the parameters of this queue earlier.In other words, ASPN-LA-PP samples from Q_1 more frequently and thus, the probabilistic priority of this queue passes the lower bound $\frac{\mu_1}{\mu_1+\mu_2}$ earlier.
- In both regions, for a fixed ρ_1, by decreasing the value of ρ_2 the value of n^*increases. To describe the reason behind this phenomenon, we have to note that when ρ_2 decreases, the number of times, when server is idle and there exists jobs in both queues simultaneously, decreases. Therefore, LA_1 has less chance of selecting actions and receiving responses from the environment.
- Therefore, it's learning time increases which results in n^* to increase as well.
- In both regions, for a fixed ρ_2, by decreasing the value of ρ_1 the value of n^*increases. This is again due to the fact that when ρ_1 decreases, the number of times, when server is idle and there exists jobs in both queues simultaneously, decreases.

The least value of n^* is obtained when both ρ_1 and ρ_2 assumes their highest values and the highest value n^* is obtained when both ρ_1 and ρ_2 assumes their least values.

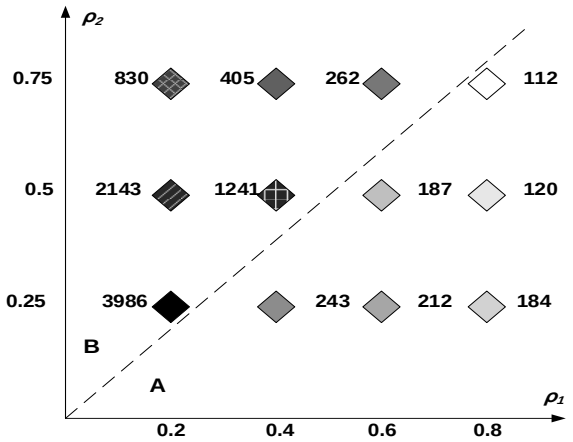

Fig. 7. The value of $\boldsymbol{n}^*$ for different queuing systems.

We repeat this experiment for all queuing systems reported in Fig. 7 and report the results in Fig. 8. In this figure, X-axis represents the values of μ_1 and μ_2 and Y-axis represents the values of $\rho_1$and ρ_2. Region A in this figure plots the results when $\rho_1 > \rho_2$ and region B plots the results when$\rho_2 > \rho_1$. In region A, Y-axis is sorted first by ρ_1 and then byρ_2, whereas in region B, Y-axis is sorted first by ρ_2 and then byρ_1. What we can see in this figure for all considered queuing systems is in coincidence with what we have concluded from Fig. 7.

E. PA Problem in a Queuing System with m Queues

The aim of the priority assignment system is to distinguish the class of job with minimum average service time. To follow this end, we extend the application of priority assignment, given in Section VI.A, to a queuing system with *m* queues and one server. Based on the proposed ASPN-LA, a new model called ASPN-LA-[m]PP is constructed and is shown in Fig. 9. To obtain the ASPN-LA-[m]PP system, the reinforcement signal generator function set $\hat{F} = \{f_1\}$ is needed. $f_1$is executed upon the firing of t_1^u and generates the reinforcement signal β_1 for LA_1. Considering the parameters $\delta(n)$and $\Gamma_i(n)$, introduced in Section B,f_1 can be described using equation (6) given below:

$$\begin{cases} \beta_1 = 1;\ \delta(n) \geq \frac{1}{\mathrm{m}} \times \sum_{\mathrm{i}} \Gamma_{\mathrm{i}}(n) \\ \beta_1 = 0;\ \delta(n) < \frac{1}{\mathrm{m}} \times \sum_{\mathrm{i}} \Gamma_{\mathrm{i}}(n) \end{cases} \quad (6)$$

where$\beta_1 = 1$ is the penalty signal and $\beta_1 = 0$is considered as a reward. Using β_1, generated according to equation (6), LA_1updates its available action probability vector, $\hat{q}(n)$, according to the $L_{R\text{-}I}$ learning algorithm described in equation (1). Then the probability vector of the actions of the chosen subset is rescaled according to equation (4).

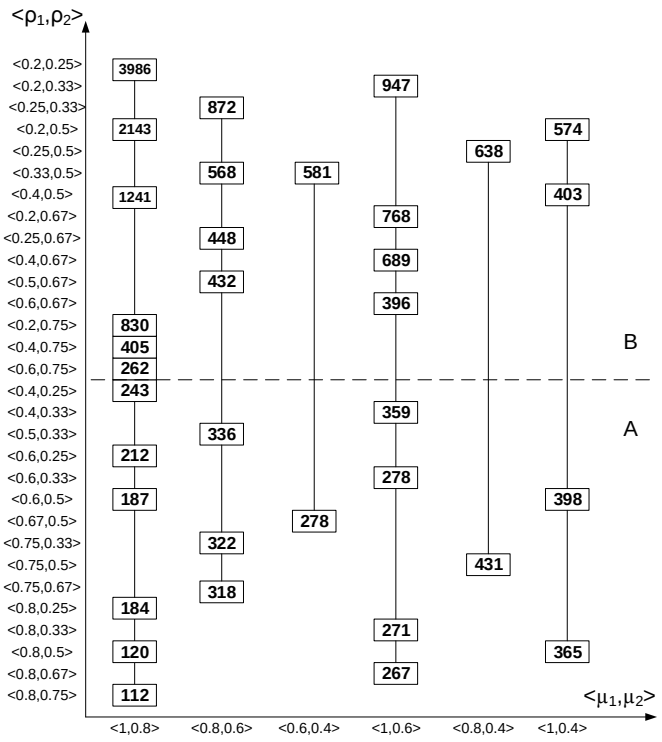

Fig. 8. The value of $\boldsymbol{n}^*$ for all queuing systems reported in TABLE I.

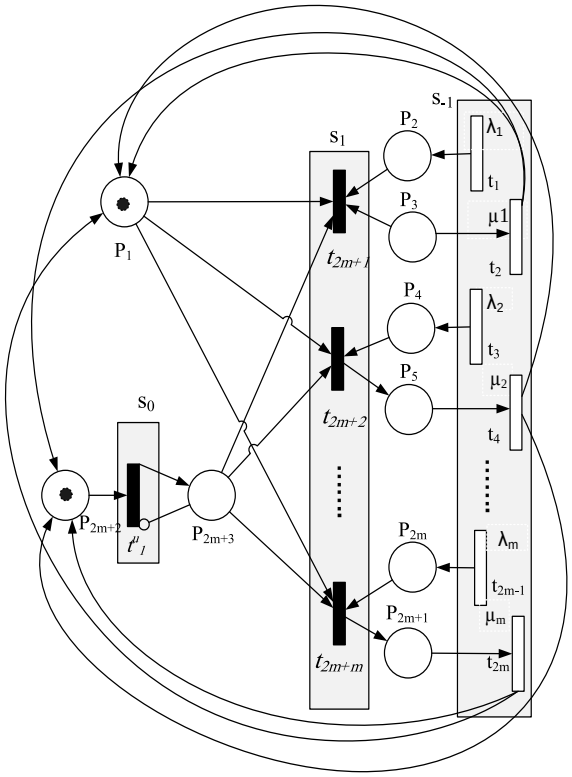

Fig. 9. ASPN-LA-[m]PP models the PA problem in a queuing system with m queues and one server.

VII. Conclusion

In this paper, we proposed a new adaptive Petri net based on learning automata called APN-LA, in which conflicts among transitions can be resolved adaptively considering the environmental changes. To this end, LA with variable number of actions can be fused with PN in order to resolve such conflicts adaptively. We also generalize the proposed APN-LA to ASPN-LA which is a fusion between LA and Stochastic Petri net (SPN). For studying the behavior of the proposed ASPN-LA, a priority assignment problem in queuing system consists of some queues with unknown parameters was defined. In this problem, we seek for a probabilistic priority assignment strategy which can selects jobs from the queues considering the average waiting time of the system. ASPN-LA was used to solve this problem. The steady-state behavior of this ASPN-LA was analyzed theoretically and it was shown that it can assign higher probabilistic priority to the queue with lower average service time. In addition, a number of computer simulations were conducted to study the behavior of ASPN-LA in this priority assignment problem in comparison to SPN. Results of these simulations showed the superiority of the proposed ASPN-LA over SPN in terms of the average waiting time of the system. As a future work, the application of LA to control the firing of timed transitions in ASPN-LA is discussed.

APPENDIX A

The finite and irreducible CTMC corresponding to the ASPN-LA-PP (Fig. 5) is shown in Fig. 6(b)where each state *j* corresponds to the tangible marking $M_j$of the *ERG*(Fig. 6(a)). Considering this CTMC, the steady-state probability vector πcan be derived using equation $\pi\mathbb{Q} = 0$ where $\mathbb{Q}$ is the infinitesimal generator matrix[50]. Let $\pi_i$denotes the steady-state probability of the CTMC being in state *i* and $P^{(i)}, i = 1,2$denotes in equation (7).

$$P^{(1)} = \pi_1 + \pi_4$$
$$P^{(2)} = \pi_3 + \pi_5 \quad (7)$$

To follow steady-state analysis to obtain $P^{(1)}$and $P^{(2)}$, we drive equation $\pi\mathbb{Q} = 0$for state 1,4,3 and 5.

$$\mu_1\pi_1 = \lambda_1\pi_4$$
$$\mu_2\pi_3 = \lambda_2\pi_5$$
$$\mu_1\pi_4 + \lambda_1\pi_4 = q_1^*\mu_1\pi_1 + q_1^*\mu_2\pi_3 + q_1^*\mu_1\pi_{18} + q_1^*\mu_2\pi_{19}$$
$$\mu_2\pi_5 + \lambda_2\pi_5 = q_2^*\mu_1\pi_1 + q_2^*\mu_2\pi_3 + q_2^*\mu_1\pi_{18} + q_2^*\mu_2\pi_{19} \quad (8)$$

By simplification of the equation (8)

$$\mu_1(\pi_1 + \pi_4) = q_1^*(\mu_1\pi_1 + \mu_2\pi_3 + \mu_1\pi_{18} + \mu_2\pi_{19})$$
$$\mu_2(\pi_3 + \pi_5) = q_2^*(\mu_1\pi_1 + \mu_2\pi_3 + \mu_1\pi_{18} + \mu_2\pi_{19})$$

and

$$\frac{(\pi_1 + \pi_4 = P^{(1)})}{(\pi_3 + \pi_5 = P^{(2)})} = \frac{q_1^*}{q_2^*} \times \frac{\mu_2}{\mu_1}$$

To select more jobs from the first queue, the value of $P^{(1)}$must be greater than $P^{(2)}$. Substituting $q_2^* = 1 - q_1^*$, the equation $q_1^* > \frac{\mu_1}{\mu_1+\mu_2}$holds. Therefore, the Theorem 1 is proved.

APPENDIX B

The lower total waiting time is obtained, when the higher priority is assigned to class of jobs with the lower average service time [46]-[47]. From learning procedure L_{R-I} shown in equation (1), if action 1 is attempted at instant *n*, the probability $q_1(n)$ is increased at instant *n+1* by an amount

proportional to $1 - q_1(k)$ for a favorable response and fixed for an unfavorable response. By this method, it follows that $\{q(n)\}_{n>0}$can be described by a Markov-process whose state space is the unit interval [0, 1], when automaton operates in an environment with penalty probabilities $\{c_1, c_2\}$. The schema L_{R-I} consists of two absorbing states $e_1 = [1,0]$and $e_2 = [0,1]$. Since the probability $q_i(n)$can decrease only when $\alpha_i$is chosen and results a favorable response, the probability $q(k) = e_i$ holds if $q(n) = e_i$ for $i = 1,2$ and all $k \geq n$. Hence $V \triangleq \{e_1, e_2\}$ represents the set of all absorbing states and the Markov process $\{q(n)\}_{n>0}$ generated by the schema L_{R-I} converges to the set V with probability one.

To study the asymptotic behavior of the process $\{q(n)\}_{n>0}$, a common method is to compute the conditional expectation of $q_1(n+1)$ given $q_1(n)$. For the schemaL_{R-I}, this computation shows that the expected value of $q_1(n)$ increases or decreases monotonically with *n* depending on whether c_2 is greater than or less than c_1. Study on the asymptotic behavior shows that $q_1(n)$ converges to *0* with a higher probability when $c_1 > c_2$ and to 1 with a higher probability when $c_1 < c_2$ if the initial probability is $q(0) = [\frac{1}{2}, \frac{1}{2}]$[13][13].

To prove Theorem 2, it is enough to show that penalty signal generated by ASPN-LA-PP system for the first class of jobs, is lower than one for the second class. In our queuing system, each class of job has a service time distribution with the exponential density function as equation (9).

$$f_i(t) = \mu_i e^{-\mu_i t} \quad (9)$$

The reinforcement signal generator function produces the penalty signal $\beta_i(n) = 1, i = 1,2$ for class *i* in instant *n* by probability value $c_i(n), \mathrm{i} = 1,2$. Let $\Gamma(n) = \frac{1}{2} \times [\Gamma_1(n) + \Gamma_2(n)]$ is the average of system waiting time to the instant *n*. So, $c_i(n)$is the probability of the service time of n^{th} jobwill more than $\Gamma(n)$ if the job is selected form queue *i*. The value of $c_i(n)$can be defined by equation (10) [46].

$$c_i(n) = prob[\delta_i(n) > \Gamma(n)] = e^{-\mu_i \Gamma(n)}, i = 1,2 \quad (10)$$

where $\delta_i(n)$ is the execution time of job *n* which is selected from i^{th} class for service. Based on our assumption, $\mu_1 > \mu_2$, it is clear that $c_1(n) < c_2(n)$ with probability one for all *n* [46]. Therefore, in ASPN-LA-PP system,$q_1(n)$ converges to 1 with probability one when *n* goes to infinity. In other words, according to Theorem 1, $q_1(n) > \frac{\mu_1}{\mu_1 + \mu_2}$ holds when *n* goes to infinity when L_{R-I} is used to update the probability vector of LA_1. Therefore, the Theorem 2 is proved.

REFERENCES

[1] W. Reisig, *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*, Springer Science & Business, 2013.

[2] T. Murata, "Petri nets: properties, analysis and applications", *Proc. IEEE*, vo1.77, pp.541-580, 1989.

[3] J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981.

[4] F. Bause, "On the analysis of Petri net with static priorities", *Acta. Informatica*, Vol. 33, pp. 669-685, 1996.

[5] F. Bause, "Analysis of Petri nets with a dynamic priority method", *Application and Theory of Petri Nets*, pp.215-234, 1997.

[6] A.Marsan, et al., M*odeling with generalized stochastic Petri net*, Wiley Series in Parallel Computing, John Wiley and Sons, 1995.

[7] K. Jensen, "Colored Petri Nets and the Invariant-Method", *Theoretical Computer Science*, Vol. 14, No. 3, pp. 317–336, 1981.

[8] H. D. Burkhard, "Ordered firing in Petri nets", *EIK (Journal of information processing and cybernetics*, vol. 17, No. 2/3, pp. 71-86, 1981.

[9] H.D. Burkhard, "Control of Petri nets by finite automata", *Fundamental Informatica (series IV)*, vol. 2, pp. 185-215, 1983.

[10] M. C. Zhou and K. Venkatesh, *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach*, World Scientific, Singapore, 1998.

[11] M. A. Thathachar and P. S. Sastry, *Networks of learning automata: Techniques for online stochastic optimization*, Springer, 2004.

[12] A. Rezvanian, et al., "Sampling from Complex Networks using Distributed Learning Automata", *Physica A: Statistical Mechanics and Its Applications*, Vol. 396, pp. 224-234, 2014

[13] K.S. Narendra, M.A.L. Thathachar, *Learning Automata: An Introduction*, Prentice-Hall, New York, 1989. M. A. L.

[14] J. A. Torkestani and M.R. Meybodi, "Finding minimum weight connected dominating set in stochastic graph based on learning automata", *Information Sciences*, No. 200, pp. 57-77, 2012.

[15] M. Esnaashari and M. R. Meybodi, "Deployment of a Mobile Wireless Sensor Network with k-Coverage Constraint: A Cellular Learning Automata Approach", *Wireless Networks*, Vol. 19, Issue 5, pp. 945-968, 2013.

[16] S. M. Vahidipour and M. R. Meybodi, "Learning in Stochastic Neural Networks using Learning Automata", *Proceedings of Information and Knowledge Technology Conference, Shiraz* University, Shiraz, 2013.

[17] F. Amiri, et al., "A novel community detection algorithm for privacy preservation in social networks", *Intelligent Informatics*, pp. 443–450, 2013.

[18] J. Zhang, C. Wang, and M. C. Zhou, "Last-Position Elimination-Based Learning Automata," *Cybernetics, IEEE Transactions on* , vol.44, no.12, pp.2484,2492, Dec. 2014.

[19] J. A. Torkestani and M. R. Meybodi, "A Learning Automata-based Cognitive Radio for Clustered Wireless Ad-Hoc Networks", *Journal of Network and Systems Management*, Vol. 19, No. 2, pp. 278-297, 2010.

[20] M. A. Thathacher and B. R. Harita, "Learning automata with changing number of actions", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 17, no. 6, pp. 1095-1100, 1987.

[21] G. Ciardo, et al., "Automated generation and analysis of Markov reward models using stochastic reward nets", *Linear Algebra, Markov Chains, and Queuing Models*, Springer New York, pp. 145-191, 1993.

[22] M. Hack, *Petri net languages*, C.S.G. Memo 124, Project MAC, M.I.T., 1975.

[23] F. Bause, "On the analysis of Petri net with static priorities", *Acta Informatica*, Vol. 33, pp. 669-685, 1996.

[24] M. Kounty, "Modeling systems with dynamic priorities", *Advances in Petri nets*, pp. 251-266, Springer-Verlag, 1992.

[25] F. Bause, "Analysis of Petri nets with a dynamic priority method", *Application and Theory of Petri Nets*, pp.215-234, 1997.

[26] H. D. Burkhard, "An investigation of controls for concurrent systems based on abstract control languages", t*heoretical Computer Science*, vol. 38, pp.193-222, 1985.

[27] J. Moodyand P. J. Antsaklis, "*Supervisory control of discrete event systems using Petri nets",* Vol. 8, Springer, 1998.

[28] G. Stremersch, "*Supervision of Petri nets",* Vol. 13. Springer, 2001.
[29] Y. Chen, Z. Li, A. Al-Ahmari, "Nonpure Petri Net Supervisors for Optimal Deadlock Control of Flexible Manufacturing Systems," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on* , vol.43, no.2, pp.252,265, March 2013
[30] L. E. Holloway, B. Krogh, and A. Giua, "A survey of Petri net methods for controlled discrete event systems", *Discrete Event Dynamic Systems*, vol. 7, no. 2, pp. 151-190, 1997.
[31] L. E. Holloway and B. H. Krogh, "Controlled Petri nets: A tutorial survey",*the11th International Conference on Analysis and Optimization of Systems Discrete Event Systems*, Springer Berlin Heidelberg, pp. 158-168, 1994.
[32] Y. Li and W. M. Wonham, "Control of vector discrete-event systems. II. Controller synthesis", *IEEE Transactions on Automatic Control*, vol. 39, no. 3, pp.512-531, 1994.
[33] B. H. Krogh, J. Magott, and L. E. Holloway, "On the complexity of forbidden state problems for controlled marked graphs", *the 30th IEEE Conference on Decision and Control*,1991.
[34] Ul. Asar, M. Zhou and R. J. Caudill, "Making Petri nets adaptive: a critical review", *Networking, Sensing and Control*, Proceedings 05 IEEE, pp. 644 – 649, 2005.
[35] L. Rutkowski and K. Cpalka, "Flexible neuro-fuzzy systems", *IEEE Transactions on Neural Networks*, Vol. 14, No. 3, pp. 554-574, 2003.
[36] R. Khosla and T. Dillon, "Intelligent hybrid multi-agent architecture for engineering complex systems", *Neural Networks*, International Conference, Vol. 4, pp. 2449 –2454, 1997.
[37] L. C. Jain and N. Martin, *Fusion of neural networks, fuzzy sets and genetic algorithms: industrial applications*, CRC press, 1999.
[38] M. G. Kadjinicolaou, M. B. E. Abdelrazik, and G. Musgrave. "Structured analysis for neural networks using Petri nets", *Circuits and Systems, Proceedings of the 33rd Midwest Symposium*, pp. 770-773, 1990.
[39] Z. Ding, Y. Zhou, and M. C. Zhou, "Modeling Self-Adaptive Software Systems With Learning Petri Nets", *Proceedings of International Conference on Software Engineering*, Hyderabad, India, pp. 464-467, May.2014.
[40] M. Chen, J. S. Ke, and J. F. Chang, "Knowledge representation using fuzzy Petri nets", *IEEE Trans. Knowledge. Data Eng.*, vol. 2, pp. 311–319, Mar. 1990.
[41] X. F. Zha, S. Y. E. Lim and S. C. Fok, "Integration of knowledge-based systems and neural networks: neuro-expert Petri net models and applications", *Robotics and Automation, Proceedings. 1998 IEEE International Conference*, Vol. 2, pp. 1423-1428, May. 1998.
[42] M. Gao, M. Zhou, X. Huang and Z. Wu, "Fuzzy reasoning Petri nets", *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* , vol.33, no.3, pp.314,324, May 2003.
[43] G. Looney, "Fuzzy Petri nets for rule-based decision making", *Systems, Man and Cybernetics, IEEE Transactions on* , vol.18, no.1, pp.178,183, Jan/Feb 1988.
[44] J. Cardoso, R. Valette, and D. Dubois, "Possibilistic Petri nets", *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on , vol.29, no.5, pp.573,582, Oct 1999*.
[45] S. Barzegar, et al., "Formalized learning automata with adaptive fuzzy coloured Petri net; an application specific to managing traffic signals", *Scientia Iranica*, Vol. 18, No. 3, pp. 554–565, 2011.
[46] M. R. Meybodi and S. Lashmivarahan, "A Learning Approach to Priority Assignment in a Two Class M/M/1 Queuing System with Unknown Parameters", *the Yale Workshop on Adaptive System Theory*, pp. 106–109, 1983.
[47] A. Cobham, "Priority Assignment in Waiting Line Problems", *Operations Research*, Vol. 2, No. 1, pp. 70–76, 1954.
[48] L. Kleinrock, *Queuing Systems*, Wiley, 1975.
[49] Y. Jiang, C. K. Tham, and C. C. Ko, "A probabilistic priority scheduling discipline for multi-service networks", *Computer Communications*, vol. 25, no. 13, pp. 1243-1254, 2002.
[50] Bolch, et. al., *Queuing System and Markov chain*, the second edition, Wiley Publication, 2006.

S. Mehdi Vahidipour received the BSc and MSc degrees in computer engineering in Iran, in 2000 and 2003, respectively. He is also a lecturer in University of Kashan.
He is currently working toward the PhD degree in computer engineering in Amirkabir University of Technology, Iran. His research interests include distributed artificial intelligence, learning automata, and Adaptive Petri nets.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in economics from Shahid Beheshti University, Tehran, Iran, in 1973 and 1977, respectively, and the M.S. and Ph.D. degrees in computer science from Oklahoma University, Norman, OK,USA, in 1980 and 1983, respectively.
He is currently a Full Professor with Computer Engineering Department, Amirkabir University of Technology, Tehran. Prior to his current position, he was an Assistant Professor with Western Michigan University, Kalamazoo, MI, USA, from 1983 to 1985, and an Associate Professor with Ohio University, Athens, OH, USA, from 1985 to 1991. His current research interests include channel management in cellular networks, learning systems, parallel algorithms, soft computing, and software development.

Mehdi Esnaashari received the B.S., M.S., and the Ph.D. degrees in computer engineering from the Amirkabir University of Technology, Tehran, Iran, in 2002, 2005, and 2011, respectively.
He is currently an Assistant Professor with Iran Telecommunications Research Center, Tehran. His current research interests include computer networks, learning systems, soft computing, and information retrieval.