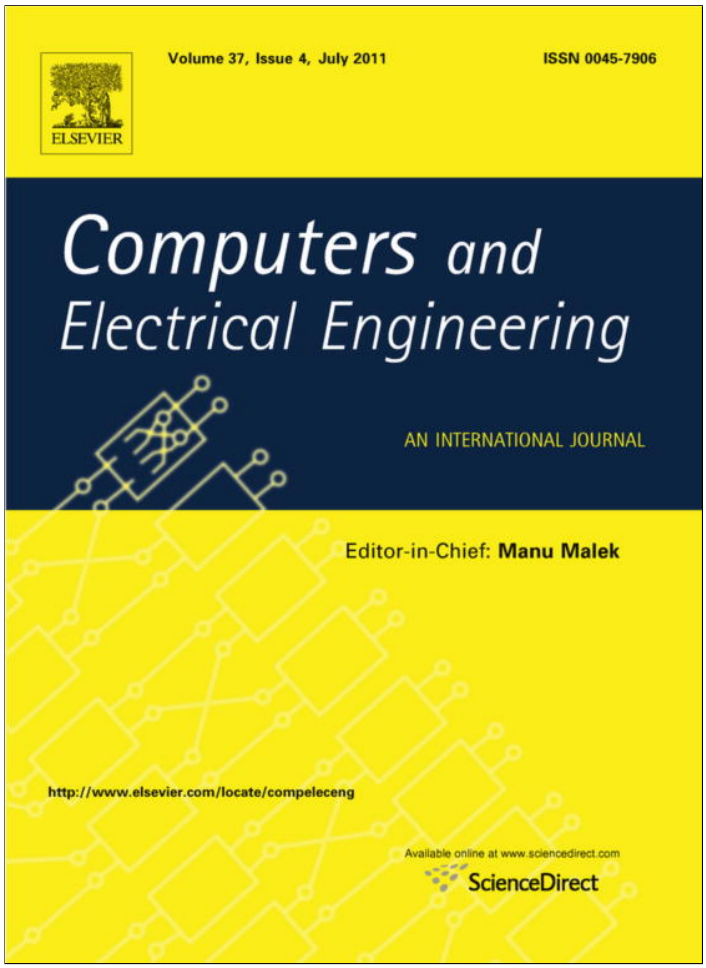

Computers and Electrical Engineering 37 (2011) 601–613

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Learning automata based dynamic guard channel algorithms ☆

Hamid Beigy [a,*], M.R. Meybodi [b]

[a] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[b] Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 14 November 2009
Received in revised form 28 March 2011
Accepted 4 April 2011
Available online 14 May 2011

ABSTRACT

In this paper, we first propose two learning automata based decentralized dynamic guard channel algorithms for cellular mobile networks. These algorithms use learning automata to adjust the number of guard channels to be assigned to cells of network. Then, we introduce a new model for nonstationary environments under which the proposed algorithms work and study their steady state behavior when they use L_{R-I} learning algorithm. It is also shown that a learning automaton operating under the proposed nonstationary environment equalizes its penalty strengths. Computer simulations have been conducted to show the effectiveness of the proposed algorithms. The simulation results show that the performances of the proposed algorithms are close to the performance of guard channel algorithm that knows all the traffic parameters.

1. Introduction

In the last decade, there is an increase in the popularity of mobile computing systems, which results in an increase for channel (bandwidth) demands. Since the number of channels allocated for this purpose is limited, cellular networks are introduced, in which the service area is partitioned into regions called *cells*. Every cell is serviced by a server called *base station*. When a mobile station moves across the cell boundary while using channels, handoff is required. If an idle channel is available in the destination cell, then the call is resumed; otherwise the call is dropped. The dropping probability of handoff calls (B_h) and the blocking probability of new calls (B_n) are important quality of service (QoS) measures of the cellular networks. Since the disconnection in the middle of a call is highly undesirable, dropping of handoff calls is more serious than blocking of new calls. Blocking more new calls generally improves the dropping probability of handoff calls and admitting more new calls generally improves the blocking probability of new calls. In order to control these QoS measures, *call admission control algorithms* are introduced, which determine whether a new call should be admitted or blocked. Both blocking probability of new calls and dropping probability of handoff calls are affected by call admission control algorithms. Several call admission algorithms have been proposed in the literature. *Fractional channel algorithm* accepts new calls with a certain probability that depends on the current channel occupancy and accepts handoff calls as long as channels are available [1]. A restricted version of this algorithm is *uniform fractional channel algorithm* (UFC), which accepts new calls with probability of π independent of channel occupancy [2]. It is shown that there is an optimal π^*, which minimizes the blocking probability of new calls with the constraint on the dropping probability of handoff calls. An algorithm for finding π^* and conditions for which the uniform fractional guard channel performs better than guard channel is given in [2]. Another restricted version of fractional channel algorithm is called *guard channel algorithm*, which reserves a subset

☆ Reviews processed and proposed for publication by Associate Editor Dr. Gill Rafael Tsouri.
* Corresponding author. Tel.: +9821 66164624.
E-mail addresses: beigy@ce.sharif.edu (H. Beigy), mmeybodi@aut.ac.ir (M.R. Meybodi).

0045-7906/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2011.04.003

of channels allocated to a cell, called *guard channels*, for handoff calls (say $C - T$ channels) [3–5], where $0 \leqslant T \leqslant C$ is a threshold and C is the number of channels allocated to the cell. Whenever the channel occupancy exceeds the threshold T, the algorithm rejects new calls until the channel occupancy goes below the threshold. The guard channel algorithm accepts handoff calls as long as channels are available. As the number of guard channels increases, the dropping probability of handoff calls will be reduced while the blocking probability of new calls will be increased [4]. It has been shown that there is an optimal threshold T^* in which the blocking probability of new calls is minimized subject to the constraint on the dropping probability of handoff calls [1]. Algorithms for finding T^* are given in [1,4,5]. If only the dropping probability of handoff calls is considered, the guard channel algorithm gives very good performance, but the blocking probability of new calls is degraded to a great extent. In order to have more control on both the dropping probability of handoff calls and the blocking probability of new calls, *limited fractional guard channel algorithm* (LFG) is proposed [1]. This algorithm, which reserves non-integral number of guard channels for handoff calls, uses an additional parameter π and operates the same as the guard channel algorithm except when T channels are occupied in the cell, in which case new calls are accepted with probability π. It has been shown that there is an optimal pair (T^*, π^*), which minimizes the blocking probability of new calls subject to the constraint on the dropping probability of handoff calls [1]. An algorithm for finding the optimal parameters is given in [1].

All of the above mentioned algorithms are useful when the input traffic is a stationary process with known parameters. Since the parameters of input traffic are unknown and possibly time varying, adaptive versions of these algorithms need to be used. In [6], two adaptive uniform fractional channel algorithm based on learning automata are introduced in which the parameter of uniform fractional guard channel algorithm is adjusted according to the traffic condition. In [7,8], dynamic guard channel algorithms are proposed in which the number of guard channels in any particular cell is adjusted based on the number of ongoing calls in neighboring cells. For more information about the call admission algorithms in cellular mobile networks, the readers may refer to [9].

In other hand learning automata are simple agents that have many desirable features such as they can be used without any priori information about the underlying application with large amount of uncertainty, require a very little and simple feedback from their environment, are very simple in structure and can be implemented easily in software or hardware, and require a few mathematical operations at each iteration so they can be used in real-time applications. Learning automata are also very useful in multi-agent and distributed systems with limited intercommunication and incomplete information and unlike traditional hill-climbing algorithms, hill-climbing in learning automata is done in expected sense in a probability space. Optimization algorithms based on learning automata do not need the objective function to be an analytical function of adjustable parameters. Learning automata have flexibility and analytical tractability needed for most applications. These features make learning automata as a useful tool for finding the near optimal number of guard channels.

In this paper, we first propose two learning automata based adaptive and autonomous call admission control algorithms. These algorithms only use the current channel occupancy of the given cell and dynamically adjust the number of guard channels for that cell. The proposed algorithms adapt the number of guard channels in such a way that the blocking probability of the new calls is minimized subject to the constraint on the dropping probability of the handoff calls. Since the learning automaton starts its learning without any priori knowledge about its environment, the proposed algorithms do not need any a priori information about input traffic and can adapt itself to the varying traffic. One of the most important advantages of the proposed algorithms is that no status information will be exchanged among the neighboring cells, although addition of such information increases the performance of the call admission algorithm [10,11]. The simulation results show that the performances of the proposed algorithms are close to the performance of guard channel algorithm that knows all traffic parameters. In this paper, we also formulated the nonstationary environment under which the proposed algorithms work and then study their behavior when they use L_{R-I} learning algorithm. It is also shown that a learning automaton operating under the proposed nonstationary environment equalizes its penalty strengths.

The rest of this paper is organized as follows. Guard channel algorithm and learning automata are given in Sections 2 and 3, respectively. The proposed adaptive call admission control algorithms are given in Section 4. The simulation results are given in Section 5 and Section 6 concludes the paper.

2. Basics in guard channel algorithm

In this section, we briefly describe the guard channel algorithm and compute its blocking performance. We assume that any given cell has a limited number of full duplex channels, C, in its channel pool. The guard channel algorithm, which is depicted algorithmically in Algorithm 1, reserves a subset of channels allocated to a particular cell for handoff calls (say $C - T$ channels) [3], where $0 \leqslant T \leqslant C$ is a threshold. These $C - T$ channels are called *guard channels*. Whenever the channel occupancy exceeds threshold T, guard channel algorithm rejects new calls until the channel occupancy goes below T. The guard channel algorithm accepts handoff calls as long as channels are available.

Algorithm 1. The algorithmic description of the guard channel algorithm

```
1:  procedure GUARDCHANNEL(C, T)
2:    if NEW CALL then                ▷ if the incoming call is a new call
3:      if c(t) < T then
4:        accept call                 ▷ if the number of occupied channels is less than the threshold
5:      else
6:        block call
7:      endif
8:    else                            ▷ if the incoming call is a handoff call
9:      if c(t) < C then
10:       accept call                 ▷ if the cell has free channels
11:     else
12:       drop call
13:     end if
14:   end if
15: end procedure
```

We consider a homogeneous cellular network where all cells have the same number of channels C and experience the same new and handoff call arrival rates. In each cell, the arrival of new calls and handoff calls are Poisson distributed with arrival rates λ_n and λ_h, respectively, and the channel holding time of new and handoff calls are exponentially distributed with mean μ^{-1}. This set of assumptions have been found reasonable as long as the number of mobile users in a cell is much greater than the number of channels allocated to that cell. We define the state of a particular cell at time T to be the number of busy channels in that cell and is represented by $c(t)$. $\{c(t)|t \geqslant 0\}$ is a continuous-time Markov chain (birth–death process) with states $0, 1, \ldots, C$. The state transition rate diagram of a cell with C full duplex channels and guard channel algorithm is shown in Fig. 1.

Define the steady state probability

$$P_n = \lim_{t\to\infty} \text{Prob}[c(t) = n \quad n = 0, 1, \ldots, C. \tag{1}$$

The steady state probability P_n that n channels are busy is given by the following expression [3].

$$P_n = \begin{cases} \frac{\rho^n}{n!} P_0 & 0 \leqslant n \leqslant T \\ \frac{\rho^n \alpha^{n-T}}{n!} P_0 & T < n \leqslant C, \end{cases} \tag{2}$$

where $\lambda = \lambda_n + \lambda_h, \alpha = \lambda_h/\lambda, \rho = \lambda/\mu$, and P_0 is the probability that all channels are free and is calculated by the following expression.

$$P_0 = \left[\sum_{k=0}^{T-1} \frac{\rho^k}{k!} + \sum_{k=T}^{C} \frac{\rho^k \alpha^{k-T}}{k!} \right]^{-1}. \tag{3}$$

Using above expressions, we can drive an expression for dropping probability of handoff calls using C channels and $C - T$ guard channels.

$$B_h(C, T) = \frac{\rho^C \alpha^{C-T}}{C!} P_0. \tag{4}$$

Similarly, the blocking probability of new calls is given by the following expression.

$$B_n(C, T) = \sum_{k=T}^{C} \frac{\rho^k \alpha^{k-T}}{k!} P_0. \tag{5}$$

3. Learning automata

The automata approach to learning involves determination of an optimal action from a set of allowable actions. An automaton can be regarded as an abstract object with finite number of actions [12]. It selects an action from its finite set

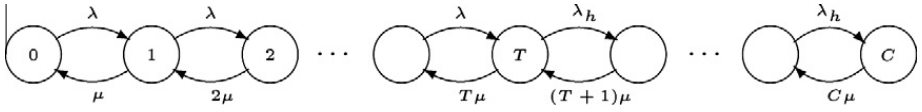

Fig. 1. Markov chain model of a cell for guard channel algorithm.

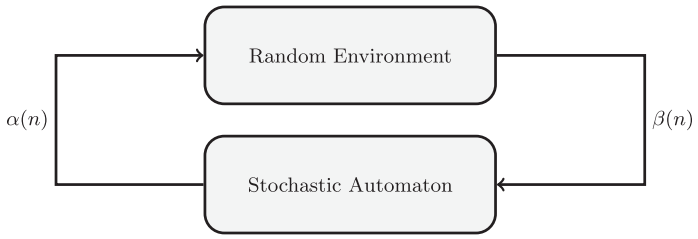

Fig. 2. The interaction of an automaton and its environment.

of actions and applies it to a random environment. The random environment evaluates the applied action and gives it a response. The response from the environment is used by automaton to modify its action probabilities (p) and to select its next action. By continuing this process, the automaton learns to select the action with the highest reward. The interaction of an automaton with its environment is shown in Fig. 2.

An automaton acting in an unknown random environment and improves its performance using a learning algorithm in some specified manner, is referred to as *learning automaton* (LA). The crucial factor affecting the performance of a learning automaton is learning algorithm. Various learning algorithms have been reported in the literature. Let α_i be the action chosen at time k as a sample realization from probability distribution $p(k)$. In linear reward-inaction algorithm the recurrence equation for updating $p_j(n)$ for $j = 1, 2, \ldots, r$ is defined as

$$p_j(n+1) = \begin{cases} p_j(n) - a[1-\beta(n)]p_j(n) & j \neq i \\ p_j(n) + a[1-\beta(n)]\sum_{k \neq i} p_k(n) & j = i, \end{cases} \tag{6}$$

where parameter $0 < a < 1$ represent *step length* that determines the amount of increase of the action probabilities, r is the number of actions for learning automaton and $0 \leqslant \beta(n) \leqslant 1$ is the response of the environment, where smaller values of $\beta(n)$ means more favorable response. If output of the environment is binary, i.e. $\beta(n) \in \{0, 1\}$, where 0 is for *reward* and 1 is for *penalty*, the environment is called P-model and the algorithm is denoted by L_{R-I}. If output of the environment takes a finite number of values in interval $[0, 1]$, the environment is called Q-model and if output of the environment lies in interval $[0, 1]$, the environment is denoted by S-model. In Q- and S-model environments the algorithm is called SL_{R-I}. Learning automata have been used successfully in many applications such as routing and call admission control in computer network [13–15], solving NP-Complete problems [16–19], capacity assignment [20,21], neural network engineering [22–26], cellular networks [6], and too many other applications [27–29] to mention a few.

4. Dynamic guard channel algorithms

In this section, we consider cellular networks with two classes of calls and introduce two learning automata based algorithms to determine the near optimal number of the guard channels when parameters λ_n, λ_h and μ are unknown and possibly time varying. In these algorithms, learning automata are used to adapt the number of guard channels as the network operates. Let $g(t)$ be the number of guard channels at time instant T which takes values in interval $[g_{\min}, g_{\max}]$, (for $0 \leqslant g_{\min} < g_{\max} \leqslant C$). In these algorithms, each base station uses one learning automaton with action set $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$, where $r = g_{\max} - g_{\min} + 1$. Selection of action α_i by learning automaton means that the base station uses $g(t) = g_{\min} + \alpha_i - 1$ guard channels. The operation of these algorithms can be described as follows. These algorithms accept handoff calls as long as the cell has free channels. When a new call arrives at a given cell, the learning automaton assigned to this cell chooses one of its actions, say α_i. If the cell has at least $g_{\min} + \alpha_i - 1$ free channels, then the call will be accepted; otherwise it will be blocked. Then the base station computes the current estimate of the dropping probability of handoff calls ($\widehat{B}_h$) and based on the result of comparison of this quantity with the specified level of QoS (p_h), the reinforcement signal is produced and the action probability vector of the learning automaton updated using a learning algorithm. The differences between the proposed algorithms are the way that they produce reinforcement signal for the learning automata and the learning algorithm used to update the action probability vector.

4.1. Dynamic guard channel algorithm I

This algorithm, which is depicted in Algorithm 2, uses an SL_{R-I} learning automaton in each cell of the network. The reinforcement signal at time instant n is generated using the following expression.

$$\beta(n) = \psi\left(\left|\widehat{B}_h - p_h\right|\right), \tag{7}$$

where $\psi : \Re \rightarrow [0, 1]$ is a projection function. The projection function $\psi(.)$ is considered to be a continuous, nondecreasing and non-negative function that maps the set of real numbers ($\Re$) into $[0, 1]$, for example $\psi(x) = x$ can be a projection function, which maps $[0, 1]$ into $[0, 1]$. The continuity of ψ is needed because the response produced by the environment is a real number in interval $[0, 1]$, the non-negativity of function ψ is needed in order to maintain the reward and penalty nature of updat-

ing, and the nondecreasing property of ψ ($\psi'(.) \geqslant 0$) is needed for preserving the relative strength of the reinforcement signal. From (7), it is obvious that when $\widehat{B}_h$ is far from p_h, then the reinforcement signal will be large, which causes the selected action of the learning automaton to be penalized. When $\widehat{B}_h$ is near to p_h, the reinforcement signal will be small and near to zero which causes the selected action of the learning automaton to be rewarded. In other words, when $\widehat{B}_h$ is greater than p_h, the chosen number of guard channels is too small and when $\widehat{B}_h$ is smaller than p_h, the number of guard channels chosen by learning automaton is large. In other words, the reinforcement signal is an indicator of the relative distance of the dropping probability of handoff calls to the predefined level of QoS.

Algorithm 2. Learning automata based dynamic guard channel algorithm I

```
1:  procedure DGC1(C)
2:    if NEW CALL then                          ▷ if the incoming call is a new call
3:      LA chooses an action and call it α_i
4:      g_i(t) ← g_min + α_i − 1.               ▷ Learning automaton determines the number of guard channels
5:      if c(t) < C − g_i(t) then
6:        accept the incoming call
7:      else
8:        block the incoming call
9:      end if
10:     β ← ψ(|B̂_h − p_h|).
11:     for k ← 1 to r
12:       if k≠i then
13:         p_k ← p_k − a[1 − β]p_k             ▷ Penalize the chosen action
14:       else
15:         p_i ← p_i + a[1 − β](1 − p_i)       ▷ Penalize the chosen action
16:       end if
17:     end for
18:   else                                      ▷ if the incoming call is a handoff call
19:     if c(t) < C then
20:       accept the incoming call
21:     else
22:         drop the incoming call
23:     end if
24:   end if
25: end procedure
```

The operation of Algorithm 2 can be described as follows: when a handoff call arrives at a cell and if the cell has at least one free channel, then the call is accepted and a free channel is assigned to it; otherwise it is dropped. When a new call arrives to a cell, the learning automaton assigned to that cell chooses one of its actions. Let α_i be the action selected by learning automaton. Then the base station calculates the number of guard channels to be used in the cell according to the following equation.

$$g(t) = g_{min} + \alpha_i - 1.$$

The incoming new call is accepted if the cell has at least $g(t)$ free channels; otherwise it is blocked. Finally an estimate of dropping probability and the reinforcement signal is computed and then the action probability vector updated accordingly. In the rest of this section, we will study the behavior of this algorithm.

4.1.1. Behavior of dynamic guard channel algorithm I

In the rest of this section, we study the steady state behavior of this algorithm. We first give a mathematical model for the nonstationary environment under which the SL_{R-I} learning automaton operates and then study the behavior of the automaton operating in this environment. For the sake of simplicity, we use an automaton, with two actions α_1 and α_2 with probability vector $\underline{p}(n) = [p_1(n), p_2(n)]^t$ at stage n. The nonstationary environment at stage n can be completely described by input set $\underline{\alpha} = \{\alpha_1, \alpha_2\}$, a continuous output set $\underline{\beta} = \{[0,1]\}$, and the penalty strengths $s_1(n)$ and $s_2(n)$, where smaller value of β means more favorable output. The penalty strengths $s_1(n)$ and $s_2(n)$ when the automaton chooses action α_i (for $i = 1,2$) is changed according to the following equations.

$$s_j(n+1) = \begin{cases} s_j(n) + \theta r_{ij}(n) & \text{with probability of } q_i(n) \\ s_j(n) + \phi r_{ij}(n) & \text{with probability of } 1 - q_i(n), \end{cases} \tag{8}$$

where $\theta r_{ij}(n)$ and $\phi r_{ij}(n)$ is the amount of changes in $s_j(n)$ (for $j = 1,2$) when the automaton chooses action α_i (for $i = 1,2$) at stage n and

$$q_i(n) = \text{Prob}[c(n) < C - g_i(n)].$$

In general, $\theta\prime_{ij}(n), \phi\prime_{ij}(n)$, and $q_i(n)$ (for $i,j = 1,2$) can be functions of $p_1(n), s_1(n), s_2(n), \lambda_n, \lambda_h, \mu$, and C, but to keep the model both simple and tractable we assume that $\theta\prime_{ij}(n), \phi\prime_{ij}(n)$, and $q_i(n)$ are fixed $\theta\prime_{ij}, \phi\prime_{ij}$, and q_i. Since $s_1(n)$ and $s_2(n)$ in (8) are in the interval $(0,1)$ and are not close to zero or one, we need to have the following assumption.

Assumption 1. $\theta\prime_{ij}(n)$ and $\phi\prime_{ij}(n)$ are $\theta\prime_{ij}$ and $\phi\prime_{ij}$, respectively, unless they lie outside of the interval $[0,1]$. That is

$$\theta\prime_{ij}(n) = \begin{cases} \theta\prime_{ij} & \text{if } 0 \leqslant s_j(n) + \theta\prime_{ij} \leqslant 1 \\ s_j(n) & \text{if } s_j(n) + \theta\prime_{ij} < 0 \\ 1 - s_j(n) & \text{if } s_j(n) + \theta\prime_{ij} > 1. \end{cases} \tag{9}$$

Similarly,

$$\phi\prime_{ij}(n) = \begin{cases} \phi\prime_{ij} & \text{if } 0 \leqslant s_j(n) + \phi\prime_{ij} \leqslant 1 \\ s_j(n) & \text{if } s_j(n) + \phi\prime_{ij} < 0 \\ 1 - s_j(n) & \text{if } s_j(n) + \phi\prime_{ij} > 1. \end{cases} \tag{10}$$

By the above assumptions, the changes in probability strengths, $\theta\prime_{ij}(n)$ or $\phi\prime_{ij}(n)$ (for $i,j = 1,2$), are fixed except when the penalty strength $s_1(n)$ and $s_2(n)$, which is given by Eq. (8), lies outside of the interval $[0,1]$. In the following two lemmas, we compute $s_1(n)$ and $s_2(n)$. Let $\tilde{p}_1(n) = \{p_1(0), p_1(1), \ldots, p_1(n-1)\}$. It is assumed that $s_1(n)$ and $s_2(n)$ are in interval $(0,1)$ and are not close to 0 or 1, i.e. $\theta_{ij}(n)$ and $\phi_{ij}(n)$ (for $i,j = 1,2$) are fixed. The resulting analysis is approximate in the sense that it is not valid when $s_1(n)$ and $s_2(n)$ are close to zero or one.

Lemma 1. *Let $\theta_{ij}(n) = q_i(n)\theta\prime_{ij}$ and $\phi_{ij}(n) = (1 - q_i(n))\phi\prime_{ij}(n)$. The expected value of penalty strength $s_2(n)$ and $s_1(n)$ are equal to*

$$\mathrm{E}[s_2(n)|\tilde{p}_1(n)] = s_2(0) + [(\theta_{12} + \phi_{12}) - (\theta_{22} + \phi_{22})] \sum_{i=0}^{n-1} p_1(i) + n[\theta_{22} + \phi_{22}], \tag{11}$$

$$\mathrm{E}[s_1(n)|\tilde{p}_1(n)] = s_1(0) + [(\theta_{11} + \phi_{11}) - (\theta_{21} + \phi_{21})] \sum_{i=0}^{n-1} p_1(i) + n[\theta_{21} + \phi_{21}]. \tag{12}$$

Proof. Computing the expectations of $s_2(n)$ on the sequence of action probabilities $\tilde{p}_1(n) = \{p_1(0), p_1(1), \ldots, p_1(n-1)\}$, we obtain

$$\begin{aligned} \mathrm{E}[s_2(n)|\tilde{p}_1(n)] = {} & \mathrm{E}[s_2(n-1)|\tilde{p}_1(n)] + p_1(n-1)q_1(n-1)\theta\prime_{12}(n-1) + (1 - p_1(n-1))q_2(n-1)\theta\prime_{22}(n-1) \\ & + p_1(n-1)(1 - q_1(n-1))\phi\prime_{12}(n-1) + (1 - p_1(n-1))(1 - q_2(n-1))\phi\prime_{22}(n-1). \end{aligned} \tag{13}$$

Suppose that $q_i(n) = q_i$ (for $n \geqslant 0$) and using Assumption 1, the above equation becomes

$$\begin{aligned} \mathrm{E}[s_2(n)|\tilde{p}_1(n)] = {} & \mathrm{E}[s_2(n-1)|\tilde{p}_1(n)] + p_1(n-1)\theta_{12} + (1 - p_1(n-1))\theta_{22} \\ & - p_1(n-1)\phi_{12} + (1 - p_1(n-1))\phi_{22} \\ = {} & p_1(n-1)[(\theta_{12} + \phi_{12}) - (\theta_{22} + \phi_{22})] + [\theta_{22} + \phi_{22}] + \mathrm{E}[s_2(n-1)|\tilde{p}_1(n)] \\ = {} & p_1(n-1)[(\theta_{12} + \phi_{12}) - (\theta_{22} + \phi_{22})] \\ & + p_1(n-2)[(\theta_{12} + \phi_{12}) - (\theta_{22} + \phi_{22})] + 2[\theta_{22} + \phi_{22}] + \mathrm{E}[s_2(n-2)|\tilde{p}_1(n)] \end{aligned} \tag{14}$$

$$= s_2(0) + [(\theta_{12} + \phi_{12}) - (\theta_{22} + \phi_{22})] \sum_{i=0}^{n-1} p_1(i) + n[\theta_{22} + \phi_{22}]. \tag{15}$$

The expected value of penalty strength $s_1(n)$ is obtained in similar way. □

Now, we are ready to study the behavior of the SL_{R-I} learning algorithm in the given nonstationary environment. In the automaton–environment connection, $(p_1(n), s_1(n), s_2(n))$ describes a discrete time continuous state Markov process with $(0,1,0)$ and $(1,0,1)$ as the absorbing states. Computing the expectations on the sequence of action probabilities $\tilde{p}_1(n) = \{p_1(0), p_1(1), \ldots, p_1(n-1)\}$, we obtain

$$\begin{aligned} \Delta p_1(n) = \mathrm{E}[p_1(n+1) - p_1(n)|\tilde{p}_1(n)] &= a[1 - p_1(n)]p_1(n)\mathrm{E}[1 - s_1(n)|\tilde{p}_1(n)] - a[1 - p_1(n)]p_1(n)\mathrm{E}[1 - s_2(n)|\tilde{p}_1(n)] \\ &= a[1 - p_1(n)]p_1(n)\mathrm{E}[s_2(n) - s_1(n)|\tilde{p}_1(n)]. \end{aligned} \tag{16}$$

We are interested in studying the equilibrium points of Eq. (16). The equilibrium points of Eq. (16) are those points that satisfy the equation $\Delta p_1(n) = 0$, where the expected changes in the probability is zero. The equilibrium points of Eq. (16) are

$$p_1(n) \to 0, \quad p_1(n) \to 1, \quad \mathrm{E}[s_2(n) - s_1(n)|\tilde{p}_1(n)] \to 0. \tag{17}$$

We assume that Eq. (16) converges to one of its equilibrium points, i.e. $\lim_{n\to\infty} \Delta p_1(n) = 0$. Two equilibrium points $p_1(n) = 0$ and $p_1(n) = 1$ correspond to the absorbing states as in a conventional SL_{R-I} learning algorithm operating in a stationary

environment. The third equilibrium point, $E[s_2(n) - s_1(n)|\tilde{p}_1(n)] \to 0$, represents an entirely different kind of behavior and convergence of $p_1(n)$ depends on the evolution of the penalty strengths, which are studied in the following lemma.

Lemma 2. *When $p_1(n)$ converges in the sense that $E[s_2(n) - s_1(n)|\tilde{p}_1(n)] \to 0$, then*

$$\frac{1}{n}\sum_{i=0}^{n-1} p_1(i) \to \frac{[(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})]}{[(\theta_{11}+\phi_{11}) + (\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) - (\theta_{12}+\phi_{12})]}. \tag{18}$$

Proof. Substituting (11) and (12) in $E[s_2(n) - s_1(n)|\tilde{p}_1(n)] \to 0$, we obtain

$$\begin{aligned} E[s_2(n) - s_1(n)|\tilde{p}_1(n)] &= [s_2(0) - s_1(0)] + n[(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})] \\ &\quad + [(\theta_{12}+\phi_{12}) - (\theta_{22}+\phi_{22}) - (\theta_{11}+\phi_{11}) + (\theta_{21}+\phi_{21})]\sum_{i=0}^{n-1} p_1(i). \end{aligned} \tag{19}$$

Since $E[s_2(n) - s_1(n)|\tilde{p}_1(n)] \to 0$, the above equation implies that

$$\frac{1}{n}\sum_{i=0}^{n-1} p_1(i) \to \frac{[(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})]}{[(\theta_{11}+\phi_{11}) + (\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) - (\theta_{12}+\phi_{12})]}. \quad \square \tag{20}$$

Substituting Eq. (18) in (11) and simplifying we get

$$\begin{aligned} E[s_2(n)|\tilde{p}_1(n)] &= s_2(0) + n\frac{[(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})][(\theta_{12}+\phi_{12}) - (\theta_{22}+\phi_{22})]}{(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) + (\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})} + n[\theta_{22}+\phi_{22}] \\ &= s_2(0) + n\frac{(\theta_{22}+\phi_{22})(\theta_{11}+\phi_{11}) - (\theta_{21}+\phi_{21})(\theta_{12}+\phi_{12})}{(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) + (\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})}. \end{aligned} \tag{21}$$

Eq. (21) must be used with caution because its validity holds only when $\theta_{ij}(n)$, $\phi_{ij}(n)$, and $q_i(n)$ are fixed and when $s_2(n)$ is not in the vicinity of zero or one. However, Eq. (21) gives the valuable information that $E[s_2(n)|\tilde{p}_1(n)]$ decreases towards zero when the second term is negative and increases towards one when the second term is positive. A similar analysis for $s_1(n)$ yields

$$\begin{aligned} E[s_1(n)|\tilde{p}_1(n)] &= s_1(0) + n\frac{[(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})][(\theta_{11}+\phi_{11}) - (\theta_{21}+\phi_{21})]}{(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) + (\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})} + n[\theta_{21}+\phi_{21}] \\ &= s_1(0) + n\frac{(\theta_{11}+\phi_{11})(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})(\theta_{12}+\phi_{12})}{(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) + (\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})}. \end{aligned} \tag{22}$$

and hence $E[s_1(n)|\tilde{p}_1(n)]$ behaves in the same way as $E[s_2(n)|\tilde{p}_1(n)]$.

Theorem 1. *The SL_{R-I} automaton operating in the nonstationary environment as defined by equation (8) equalizes the expected penalty strengths of two actions.*

Proof. It is seen from (18) that the limit to which $\frac{1}{n}\sum_{i=0}^{n-1} p_1(i)$ converges depends only on the changes in the penalty strengths. In (18), $(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})$ corresponds to the difference between changes in s_2 and s_1 when action α_2 is performed and $(\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})$ corresponds to the difference between changes in s_1 and s_2 when action α_1 is performed. From (18), it is clear that the ratio in which the two actions α_1 and α_2 are chosen in the long run is inversely proportional to these changes, i.e. $(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})$ and $(\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})$. It must be noted that $p_1(n)$ does not converge to a fixed value since this would imply an absorbing state other than zero or one. The convergence is in the sense indicated in (19) and is that of a sample average. The probabilities of $p_1(n)$ and $p_2(n)$ of the two actions as well as the penalty strengths $s_1(n)$ and $s_2(n)$ vary with time in such manner that

$$E[s_2(n) - s_1(n)|\tilde{p}_1(n)] \to 0, \tag{23}$$

or the expected penalty strengths of two actions tend to be equalized. This is achieved by α_1 being chosen a fraction $\frac{(\theta_{22}+\phi_{22})-(\theta_{21}+\phi_{21})}{(\theta_{11}+\phi_{11})+(\theta_{22}+\phi_{22})-(\theta_{21}+\phi_{21})-(\theta_{12}+\phi_{12})}$ of the total number of times in the long run. The equalization of the expected values of penalty strengths is one significant feature of this model. $\square$

Remark 1. Let $c(n)$ be the number of busy channels at stage n, $P_{c(n)}(n)$ be the probability of having $c(n)$ busy channels and $g_i(n)$ be the number of guard channels chosen by the learning automaton at stage n. Let $\pi_{c(n)}(n)$ be the probability of accepting new calls at stage n when the cell has $c(n)$ busy channels. It is evident that $\pi_i(n)$ is a function of $P_{c(n)}(n)$ and $p_{g_i(n)}(n)$. Since $\pi_i(n)$ is updated as a result of learning, the proposed algorithm can be considered as an adaptive version of fractional guard channel algorithm.

Remark 2. The equalization of the expected values of penalty strengths is one significant feature of the proposed model. When the expected values of penalty probabilities are equalized, new calls are rejected with the same probability at different states of birth–death process model of the cell. This is similar to the adaptive uniform fractional channel algorithm [6].

Remark 3. From Remarks 2 and 1, it can be concluded that the given algorithm exhibits the behavior of adaptive fractional guard channel algorithm in transient state and adaptive uniform fractional channel algorithm in steady state.

4.2. Dynamic guard channel algorithm II

Although the blocking probability of new calls for algorithm I is lower than the blocking probability of the guard channel algorithm, but it can not maintain the predefined level of QoS, as evidenced by the results of simulation. The algorithm presented in this section, (Algorithm 3) tries to minimize the blocking probability of new calls and at the same time to maintain the specified level of QoS. This algorithm uses an L_{R-I} learning automaton in each cell for determination of the number of guard channels. The selected action of learning automaton in a cell will be rewarded if the incoming new call is accepted and the current estimate of dropping probability of handoff calls ($\widehat{B}_h$) is less than the specific level of QoS (p_h) or the incoming new call is rejected and the current estimate of dropping probability of handoff calls is greater than the specific level of QoS; the selected action neither rewarded nor penalized otherwise.

Algorithm 3. Learning automata based dynamic guard channel algorithm II

```
procedure DGC II(C)
  if NEW CALL then                                   ▷ if the incoming call is a new call
    LA chooses an action and call it α_i
    g_i(t) ← g_min + α_i − 1.                        ▷ Learning automaton determines the number of guard channels
    if c(t) < C − g_i(t) then
      accept the incoming call
    else
      block the incoming call
    end if
    B̂_h ← thenumberofdroppedhandoffcalls / totalnumberofhandoffcalls   ▷ Estimate the dropping probability of handoff calls
    if the incoming new call is accepted then
      if B̂_h ⩽ p_h then
        for k ← 1 to r do
          if k≠i then
            p_k ← (1 − a)p_k
          else
            p_i ← p_i + a(1 − p_i)
          end if
        end for
      end if
    else if B̂_h > p_h then
      for k ← 1 to r do
        if k≠i then
          p_k ← (1 − a)p_k
        else
          p_i ← p_i + a(1 − p_i)
        end if
      else for
    end if
  else
    if c(t) < C then
      accept the incoming call
    else
      drop the incoming call
    end if
  end if
end procedure
```

The operation of this algorithm can be described as follows: when a handoff call arrives at a cell, if the cell has at least one free channel, the call will be accepted and a free channel will be assigned to it; otherwise it will be dropped. When a new call arrives at a cell, the learning automaton associated to that cell will choose one of its actions. Let α_i be the action selected by learning automaton. Then the base station calculates the number of guard channels to be used in the cell according to the following equation.

$$g_i(t) = g_{min} + \alpha_i - 1.$$

The incoming new call will be accepted if the cell has at least $g_i(t)$ free channels; otherwise it will be blocked. Finally an estimate of dropping probability ($\widehat{B}_h$) is computed and if the new call is accepted and $\widehat{B}_h$ is less than p_h then the selected action of learning automaton will be rewarded; if the new call is blocked and $\widehat{B}_h$ is greater than p_h then the selected action of learning automaton will be rewarded. In the rest of this section, we study the behavior of this algorithm.*4.2.1. Behavior of dynamic guard channel algorithm II*

In what follows, first we study the behavior of this algorithm with respect to the channel utilization and then investigate the asymptotic behavior of this algorithm. This analysis is similar to analysis of the asymptotic behavior of algorithm I.

4.2.2. The channel utilization for algorithm II

Define m_n $(m_1 < m_2 < \ldots)$ be a sequence of random variables, where m_n is the time at which the n^{th} new call arrives. We have

$$\Delta m_n = m_{n+1} - m_n = 1/\lambda_n. \tag{24}$$

Let $g_i(n)$ be the number of guard channels reserved for the given cell at time m_n and $X_i(n)$ be a random variable that at time m_n takes value 1 when a new call is accepted and value 0 when a new call is rejected. Suppose that some handoff calls will arrive in time interval (m_n, m_{n+1}), thus the excepted number of handoff calls in interval (m_n, m_{n+1}) is equal to λ_h/λ_n. Suppose that the dropping probability of handoff calls in the interval $(m_n, m_{n+1}]$ to be fixed. When the n^{th} new call arrives to a cell, the learning automaton associated to that cell chooses one of its actions, say α_i. If the new call is accepted at time m_n, the expected number of busy channels at time m_{n+1} can be computed as follows.

$$\begin{aligned}\mathrm{E}[c(n+1)|X_i(n)=1] &= c(n) + 1 - [c(n)+1]\frac{\mu}{\lambda_n} + \left[1 - \mathrm{E}\left[\widehat{B}_h(n)|X_i(n)=1\right]\right]\frac{\lambda_h}{\lambda_n} \\ &= [c(n)+1]\left[1-\frac{\mu}{\lambda_n}\right] + \left[1 - \mathrm{E}\left[\widehat{B}_h(n)|X_i(n)=1\right]\right]\frac{\lambda_h}{\lambda_n},\end{aligned} \tag{25}$$

where $\left[1 - \mathrm{E}\left[\widehat{B}_h(n)|X_i(n)=1\right]\right]\frac{\lambda_h}{\lambda_n}$ is the expected number of the accepted handoff calls and $[c(n)+1]\frac{\mu}{\lambda_n}$ is the excepted number of calls that are departed from the system in interval (m_n, m_{n+1}). Similarly, if the new call is rejected at time m_n, then the expected number of busy channels at m_{n+1} can be computed as follows.

$$\begin{aligned}\mathrm{E}[c(n+1)|X_i(n)=0] &= c(n) - c(n)\frac{\mu}{\lambda_n} + \left[1 - \mathrm{E}\left[\widehat{B}_h(n)|X_i(n)=0\right]\right]\frac{\lambda_h}{\lambda_n} \\ &= c(n)\left[1-\frac{\mu}{\lambda_n}\right] + \left[1 - \mathrm{E}\left[\widehat{B}_h(n)|X_i(n)=0\right]\right]\frac{\lambda_h}{\lambda_n},\end{aligned} \tag{26}$$

where $\left[1 - \mathrm{E}\left[\widehat{B}_h(n)|X_i(n)=0\right]\right]\frac{\lambda_h}{\lambda_n}$ is the expected number of the accepted handoff calls and $c(n)\frac{\mu}{\lambda_n}$ is the excepted number of calls that are departed from the system in interval (m_n, m_{n+1}). Let $q_i(n) = \mathrm{Prob}[X_i(n)=1]$. The expected number of busy channels at time m_{n+1} is equal to

$$\begin{aligned}\mathrm{E}[c(n+1)] &= \sum_{i=1}^{r}\{\mathrm{E}[c(n+1)|X_i(n)=1]\mathrm{Prob}[X_i(n)=1] + \mathrm{E}[c(n+1)|X_i(n)=0]\mathrm{Prob}[Xi(n)=0]\}p_i(n) \\ &= \sum_{i=1}^{r}\{\mathrm{E}[c(n+1)|X_i(n)=1]q_i(n) + \mathrm{E}[c(n+1)|X_i(n)=0](1-q_i(n))\}p_i(n),\end{aligned} \tag{27}$$

where $p_i(n)$ is the probability of choosing action α_i by the automaton and r is the number of actions for the automaton.

4.2.3. The asymptotic behavior of algorithm II

In the rest of this section, we study the asymptotic behavior of algorithm II. The mathematical model for the nonstationary environment under which the L_{R-I} learning automaton operates is similar to the nonstationary environment for algorithm I. For the sake of simplicity, the analysis is done for an automaton with two actions α_1 and α_2 with probability vector $\underline{p}(n) = [p_1(n), p_2(n)]^t$ at stage n. The nonstationary environment at stage n can be completely described by input set $\underline{\alpha} = \{\alpha_1, \alpha_2\}$, a binary output set $\underline{\beta} = \{0, 1\}$, and penalty probabilities $c_1(n)$ and $c_2(n)$. The penalty probabilities $c_1(n)$ and $c_2(n)$ when the automaton chooses action α_i (for $i = 1, 2$) is changed according to the following equation.

$$c_j(n+1) = \begin{cases} c_j(n) + \theta r_{ij}(n) & \text{with probability of } q_i(n) \\ c_j(n) + \phi r_{ij}(n) & \text{with probability of } 1 - q_i(n), \end{cases} \tag{28}$$

where $\theta\prime_{ij}(n)$ and $\phi\prime_{ij}(n)$ is the amount of changes in $c_j(n)$ (for $j = 1,2$) when the automaton chooses action α_i (for $i = 1,2$) at stage n and

$$q_i(n) = \text{Prob}[c(n) < C - g_i(n)].$$

In general, $\theta\prime_{ij}(n), \phi\prime_{ij}(n)$, and $q_i(n)$ (for $i,j = 1,2$) can be functions of $p_1(n), c_1(n), c_2(n), \lambda_n, \lambda_h, \mu$, and C, but to keep the model both simple and tractable we assume that $\theta\prime_{ij}(n)$, $\phi\prime_{ij}(n)$, and $q_i(n)$ are fixed $\theta\prime_{ij}$, $\phi\prime_{ij}$, and q_i. Since $c_1(n)$ and $c_2(n)$in (28) are in the interval $[0,1]$, the Assumption 1 will be held for $\theta\prime_{ij}(n)$ and $\phi\prime_{ij}(n)$ (for $i,j = 1,2$). From the analysis done for algorithm I, the following results are immediate for algorithm II.

Lemma 3. *Let $\theta_{ij}(n) = q_i(n)\theta'_{ij}$ and $\phi_{ij}(n) = (1 - q_i(n))\phi'_{ij}(n)$. When $p_1(n)$ converges in the sense that $\mathrm{E}[c_2(n) - c_1(n)|\tilde{p}_1(n)] \to 0$, then the expected value of penalty probabilities $c_2(n)$ and $c_1(n)$ are equal to*

$$\mathrm{E}[c_2(n)|\tilde{p}_1(n)] = c_2(0) + n\frac{(\theta_{22}+\phi_{22})(\theta_{11}+\phi_{11}) - (\theta_{21}+\phi_{21})(\theta_{12}+\phi_{12})}{(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) + (\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})}, \tag{29}$$

$$\mathrm{E}[c_1(n)|\tilde{p}_1(n)] = c_1(0) + n\frac{(\theta_{11}+\phi_{11})(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21})(\theta_{12}+\phi_{12})}{(\theta_{22}+\phi_{22}) - (\theta_{21}+\phi_{21}) + (\theta_{11}+\phi_{11}) - (\theta_{12}+\phi_{12})}. \tag{30}$$

Theorem 2. *The L_{R-I} automaton operating in the nonstationary environment as defined by Eq. (28) equalizes the expected penalty strengths of its actions.*

5. Numerical example

In order to evaluate the proposed dynamic guard channel algorithms, computer experiments are conducted. In these experiments, we also compare the performance of the proposed algorithms (DGC I and DGC II) with the performance of the related algorithms: the limited fractional guard channel (LFG) [1], the uniform fractional channel (UFC) [2], the guard channel (GC) [3], and the adaptive uniform fractional channel (AUFC) [6]. Simulations are based on a single cell of a homogeneous cellular network system in which each cell has 8 full duplex channels ($C = 8$). We assume that the arrival of new calls is Poisson process with rate λ_n fixed at 30 calls per minute and the arrival of handoff calls is Poisson process with rate λ_h varied between 2 and 20 calls per minute. We also assume that the duration of calls are exponentially distributed with mean $\mu = 1/6$ and the desired level of QoS (dropping probability of handoff calls) is less than 0.01, i.e. $p_h = 0.01$ and ψ for computing the reinforcement signal of the dynamic guard channel I is set to $\psi(x) = x$. The optimal number of guard channels for guard channel and limited fractional guard algorithms are obtained by the algorithms given in [1,5] and the optimal parameters of uniform fractional channel algorithm is obtained by algorithm given in [2]. Algorithm DGC I uses the S-model environment while DGC II uses P-model environment. These simulations show that the learning automaton learns in P-model environments more quicker than S-model environments in the proposed algorithms. This is due to the fact that P-model environment uses a larger steps for reward and punishment of the chosen action. Hence, the DGC II algorithm finds the optimal action in the smaller number of iterations.

The results of simulations are given in Figs. 3 through 6. The dropping probability of handoff calls and the blocking probability of new calls for the above mentioned algorithms are shown in Figs. 3 and 4, respectively. These results are obtained by

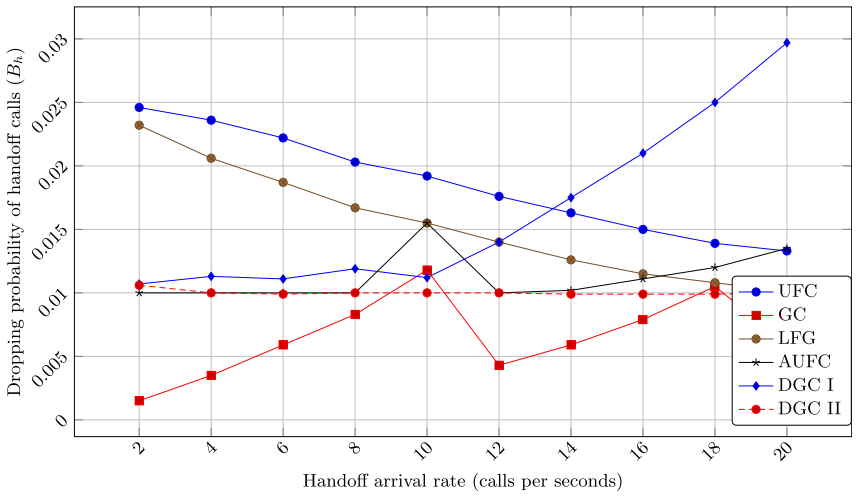

Fig. 3. The dropping probability of different algorithms.

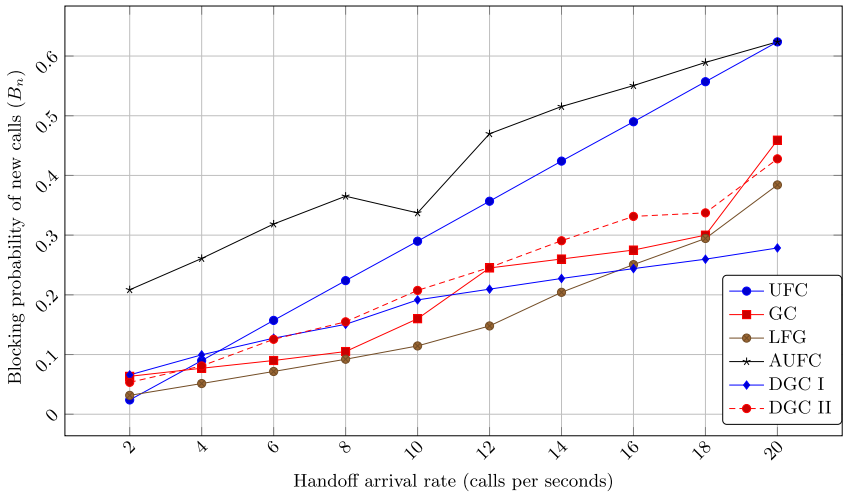

Fig. 4. The blocking probability of different algorithms.

averaging 10 runs from 2,000,000 seconds simulation of each algorithm. Fig. 3 shows that algorithm I cannot maintain the specified level of QoS, which is set to 0.01. This is because of the way that the reinforcement signal is generated. But algorithm II maintains the specified level of QoS for all handoff rates. Algorithm DGC I uses the S-model environment while DGC II uses P-model environment. These simulations show that the learning automaton learns in P-model environments more quicker than S-model environments in the proposed algorithms. This is due to the fact that P-model environment uses a larger steps for reward and punishment of the chosen action. Hence, the DGC II algorithm finds the optimal action in the smaller number of iterations.

By inspecting Fig. 4, it is evident that the performance of algorithm I (II) is close to the performance of guard channel algorithm in low (high) traffic. One reason for the difference in performances of the guard channel algorithm and the proposed algorithms is due to the transient behavior of the proposed algorithm. Since, the performance parameters (the blocking probability of new calls and the dropping probability of handoff calls) in the early stages of simulation are far from their desire value, they affect the long-time calculation of the performance parameters. However, such effect can be removed by excluding the transient behaviors of the proposed algorithm, which is shown in Figs. 5 and 6. Fig. 5 shows the evolution of the dropping probability of handoff calls for the guard channel algorithm and dynamic guard channel algorithm II. Fig. 6 shows the evolution of the blocking probability of new calls for the guard channel algorithm and dynamic guard channel algorithm II. The traffic parameters used for these figures corresponds to case 10 in Figs. 3 and 4. By carefully inspecting these figures and ignoring the transient behavior of the proposed algorithm, it can be concluded that the dropping probability of handoff calls approaches its prescribed value (p_h), while the blocking probability of new calls is less than the corresponding performance parameter of guard channel algorithm.

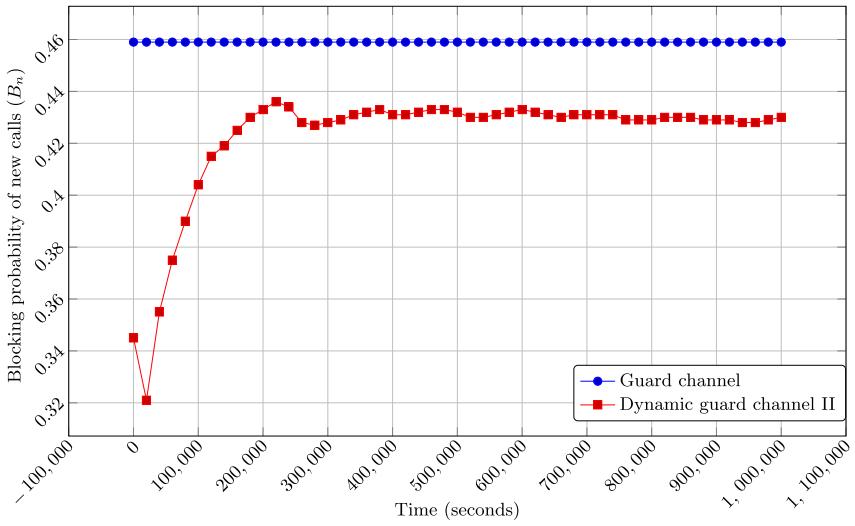

Fig. 5. The dropping probability of handoff for guard channel and dynamic guard channel II algorithms.

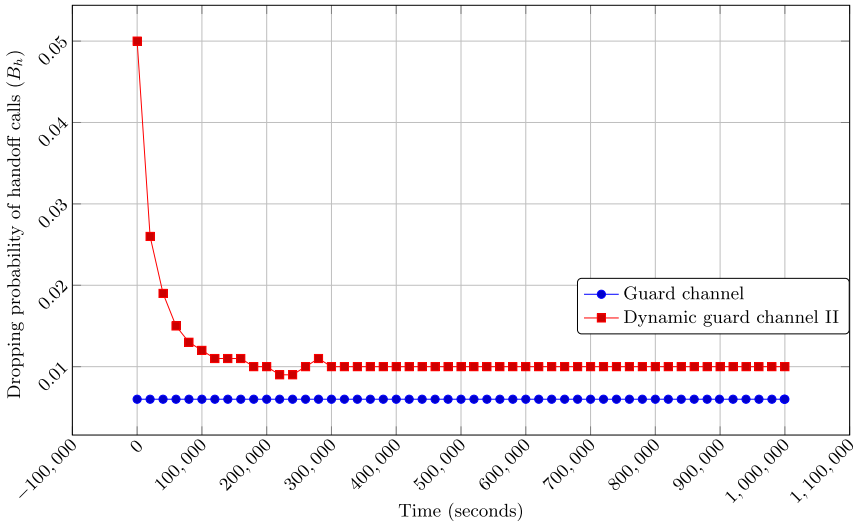

Fig. 6. The blocking probability of new calls for guard channel and dynamic guard channel II algorithms.

6. Conclusions

In this paper, we studied two decentralized dynamic guard channel algorithms which use learning automata to adjust the number of guard channels. We also introduced a new model for nonstationary environments, which describes the behavior of the proposed algorithms, and studied the steady state behavior of L_{R-I} learning algorithm operating under the proposed environment. It was shown that the learning automaton equalizes its penalty strengths. The first advantage of the proposed algorithms is that they do not need traffic parameters of the network and they adapt to changing traffic parameters. The second advantage of these algorithms is that they are decentralized and do not need any information exchange among the neighboring cells. We also studied the behavior of the proposed algorithms through computer simulations. The simulation results showed that the performances of the proposed algorithms were close to the performance of guard channel algorithm that knows all the traffic parameters. The following points can be concluded from the simulation results.

1. The proposed algorithms are able to operate in networks with unknown traffic parameters.
2. The transient response of algorithms show that the proposed algorithms follow the guard channel algorithm which knows all traffic parameters.
3. The steady state response of the algorithms shows that the proposed algorithms can maintain the dropping probability of handover calls and the blocking probability of new calls at the specified level. Although both guard channel and dynamic guard channel II provide almost the same dropping probabilities for handover calls, the proposed algorithm provides the lower blocking probability of new calls.
4. The proposed algorithms do not need any information exchange among neighboring cells and are useful in environments with unknown and incomplete information.

References

[1] Ramjee R, Towsley D, Nagarajan R. On optimal call admission control in cellular networks. Wireless Netw 1997;3:29–41.
[2] Beigy H, Meybodi MR. A new fractional channel policy. J High Speed Netw 2004;13:25–36.
[3] Hong D, Rappaport S. Traffic modelling and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoffs procedure. IEEE Trans Vehicular Technol 1986;35:77–92.
[4] Oh S, Tcha D. Prioritized channel assignment in a cellular radio network. IEEE Trans Commun 1992;40:1259–69.
[5] Haring G, Marie R, Puigjaner R, Trivedi K. Loss formulas and their application to optimization for cellular networks. IEEE Trans Vehicular Technol 2001;50:664–73.
[6] Beigy H, Meybodi MR. Adaptive uniform fractional channel algorithms. Iran J Elect Comput Eng 2004;3:47–53.
[7] Yu O, Leung V. Self-tuning prioritized call handling mechanism with dynamic guard channels for mobile cellular systems. In: Proceedings of IEEE vehicular technology conference; 1996. p. 1520–4.
[8] Peha JM, Sutivong A. Admission control algorithms for cellular systems. Wireless Netw 2001;7:117–25.
[9] Beigy H, Meybodi MR. User based call admission control policies for cellular mobile systems: a survey. CSI J Comput Sci Eng 2003;1(2(b)):45–58.
[10] Beigy H, Meybodi MR. Asynchronous cellular learning automata. Automatica 2008;44:1350–7.
[11] Beigy H, Meybodi MR. Cellular learning automata with multiple learning automata in each cell and its applications. IEEE Trans Syst Man Cybernet B Cybernet 2010;40:54–65.
[12] Narendra KS, Thathachar KS. Learning automata: an introduction. New York: Prentice-Hall; 1989.
[13] Nedzelnitsky OV, Narendra KS. Nonstationary models of learning automata routing in data communication networks. IEEE Trans Syst Man Cybernet 1987;SMC-17:1004–15.
[14] Obaidat MS, Papadimitriou GI, Pomportsis AS, Laskaridis HS. Learning automata-based bus arbitration for shared-medium ATM switches. IEEE Trans Syst Man Cybernet B Cybernet 2002;32:815–20.

[15] Papadimitriou GI, Obaidat MS, Pomportsis AS. On the use of learning automata in the control of broadcast networks: a methodology. IEEE Trans Syst Man Cybernet B Cybernet 2002;32:781–90.
[16] Oommen BJ, de St. Croix EV. Graph partitioning using learning automata. IEEE Trans Comput 1996;45:195–208.
[17] Beigy H, Meybodi MR. Solving the Graph Isomorphism Problem Using Learning Automata. In: Proceedings of the 5th annual international computer society of iran computer conference, CISCC-2000, Tehran, Iran. p. 402–15.
[18] Beigy H, Meybodi MR. Utilizing distributed learning automata to solve stochastic shortest path problems. Int J Uncertain Fuzz Knowledge Based Syst 2006;14:591–615.
[19] Granmo OC, Oommen BJ, Myrer SA, Olsen MG. Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. IEEE Trans Syst Man Cybernet B Cybernet 2007;37:166–75.
[20] Oommen BJ, Roberts TD. Continuous learning automata solutions to the capacity assignment problem. IEEE Trans Comput 2000;49:608–20.
[21] Oommen BJ, Roberts TD. Discretized learning automata solutions to the capacity assignment problem for prioritized networks. IEEE Trans Syst Man Cybernet B Cybernet 2002;32:821–31.
[22] Meybodi MR, Beigy H. Neural network engineering using learning automata: determining of desired size of three layer feedforward neural networks. J Fac Eng 2001;34:1–26.
[23] Beigy H, Meybodi MR. Backpropagation algorithm adaptation parameters using learning automata. Int J Neural Syst 2001;11:219–28.
[24] Meybodi MR, Beigy H. New learning automata based algorithms for adaptation of backpropagation algorithm parameters. Int J Neural Syst 2002;12:45–68.
[25] Meybodi MR, Beigy H. A note on learning automata based schemes for adaptation of BP parameters. J Neurocomput 2002;48:957–74.
[26] Beigy H, Meybodi MR. A learning automata-based algorithm for determination of the number of hidden units for three layer neural networks. Int J Syst Sci 2009;40:101–18.
[27] Sastry PS, Nagendra GD, Manwani N. A team of continuous-action learning automata for noise-tolerant learning of half-spaces. IEEE Trans Syst Man Cybernet B Cybernet 2010;40:19–28.
[28] Oommen BJ, Hashem MK. Modeling a student classroom interaction in a tutorial-like system using learning automata. IEEE Trans Syst Man Cybernet B Cybernet 2010;40:29–42.
[29] Ming-Feng Y, Cheng-Hung T. Standalone CMAC control system with online learning ability. IEEE Trans Syst Man Cybernet B Cybernet 2010;40:43–53.

Hamid Beigy received the B.S. and M.S. degrees in computer engineering from the Shiraz University in Iran, in 1992 and 1995, respectively. He also received the Ph.D. degree in computer engineering from the Amirkabir University of technology in Iran, in 2004. Currently, he is an associate professor in computer engineering department at the Sharif University of technology, Tehran, Iran. His research interests include, channel management in cellular networks, learning systems, parallel algorithms, and soft computing.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degree from 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, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an assistant professor at Western Michigan University, and from 1985 to 1991 as an associate professor at Ohio University, USA. His research interests include, channel management in cellular networks, 1earning systems, parallel algorithms, soft computing and software development.