# Call Admission Control in Cellular Networks: A Reinforcement Learning Solution

Sidi-Mohammed Senouci[*], André-Luc Beylot[**], Guy Pujolle[*]

[*]Laboratoire LIP6
Université de Paris VI
8, rue du Capitaine Scott
75015 Paris – France
Sidi-Mohammed.Senouci@lip6.fr
Guy.Pujolle@lip6.fr

[**]ENSEEIHT - IRIT/TeSA Lab
2, rue C. Camichel - BP7122
F-31071 Toulouse Cedex 7 – France
andre-luc.beylot@enseeiht.fr

*Abstract* – **We address in this paper, the call admission control (CAC) problem in a cellular network that handles several classes of traffic with different resource requirements. The problem is formulated as a Semi-Markov Decision Process (SMDP) problem. We use a real-time Reinforcement Learning (RL) [neuro-dynamic programming (NDP)] algorithm to construct a dynamic call admission control policy. We show that the policies obtained using our TQ-CAC and NQ-CAC algorithms, which are two different implementations of the RL algorithm, provide a good solution and are able to earn significantly higher revenues than classical solutions such as Guard Channel. A large number of experiments illustrates the robustness of our policies and shows how they do improve Quality of Service (QoS) and reduce call-blocking probabilities of handoff calls even with variable traffic conditions.**

## 1 Introduction

The increasing demand and rapid growth of mobile communications that will provide reliable voice and data communications, anytime and anywhere has massively grown. But, compared to the wire-line networks, several problems such as call admission control, channel allocation, location management and routing are more difficult to solve and owe their complexity to the shortcomings of the wireless medium. The basic goal is to maximize the utilization of a set of network resources which are both scarce and expensive.

The service area in these networks is partitioned into cells. Each cell is assigned a set of channels[1]. As a user moves from one cell to another, his call requires at least one channel to be allocated in the destination cell if is to remain active. This event (handoff) must be transparent to the user. If the destination cell has no available

channels, the call is aborted. Disconnecting ongoing calls is highly undesirable and one of the goals of the network designer is to keep low the handoff blocking probability. In [1,2,3], the authors show that the well-known Guard Channel policy, which reserves a set of channels for handoff calls, is optimal for minimizing this performance criterion. This technique is simple to dimension in a mono-class traffic framework, however the optimization is quite complicated in a multi-class context. In such a context, it is sometimes preferable to block a call of a less valuable class and to accept another call of a more valuable class. Furthermore, these techniques ignore completely the experience or knowledge that could be gained during the real-time operation of the system. In this new context, the use of learning techniques can lead to good solutions in reasonable times. A number of researchers have recently explored the application of learning algorithms like Reinforcement Learning (RL), MultiLayer Perceptron (MLP), and Genetic algorithms in telecommunications systems. In [4,5], the authors propose an alternative approach to solving the dynamic channel assignment problem through Reinforcement Learning. Other works investigate how to use learning algorithms to solve admission control [6,7,8,9] and network routing [10,11] in these systems.

This paper proposes an alternative approach to solve the call admission control (CAC) in multimedia cellular networks. The optimal CAC policy is obtained through a form of reinforcement learning algorithm known as Q-learning [12,13,14]. Instead of relying on a known teacher, the system is designed to learn an optimal assignment policy through direct interaction with the environment. We compare our policies to the guard channel scheme defined above, and to the greedy policy scheme (Policy that always accepts a call if the capacity constraint will not be violated by adding this call). The performance of the algorithms has been evaluated on the basis of three QoS metrics: total rewards of the accepted calls (*Total rewards*), total lost rewards of the rejected calls (*Total lost rewards*), and handoff blocking probabilities. In this paper, we only consider congestion control based on the dropped blocked calls discipline, the resource access priority for handoff calls may be further increased by employing the blocked queued calls discipline [1].

We consider a system with two classes of traffic. A reward (payoff) representing the cost of serving a user is associated to each class of traffic. These rewards depend also on the whether the call is new or a handoff. Our objective is to accept or reject customers so as to maximize the expected value of the rewards received over an infinite planning horizon. By making the assumptions of Poisson arrivals and a common exponential service time, this problem can be formulated as a Semi-Markov Decision Process (SMDP) and learning is a solution for this problem.

---

[1] Channels could be frequencies, time slots or codes depending on the radio access technique used.

The rest of the paper is organized as follows. After a brief description of the Q-Learning strategy, we formulate in section 2 the CAC problem as an SMDP, then we will detail in section 3 the two different implementations of the Q-Learning algorithm (TQ-CAC, and NQ-CAC) which solve the SMDP. In section 4, we present an analysis of the guard channel method. Next, section 5 exposes a performance evaluation and some numerical results. Finally, section 6 summarizes the main contributions of this work.

## 2 Problem Description

We propose an alternative approach to solving the call admission control problem. This approach is based on the idea that CAC can be regarded as an SMDP and in which learning is one of the effective ways to find a solution to this problem. A particular learning paradigm known as neuro-dynamic programming (NDP) [reinforcement learning (RL)] has been adopted. In NDP, as shown in Fig. 1, an agent aims to learn an optimal control policy by repeatedly interacting with the controlled environment in such a way that its performance, evaluated by the sum of rewards (payoff) obtained from the environment, is maximized. There exist a variety of RL algorithms and Q-learning has been found to be particularly suitable for the CAC task. In what follows, we briefly describe this algorithm (see [13,14] for more information), and then present the details of how the CAC problem can be solved by means of Q-learning.

### 2.1 Q-Learning Strategy

Assume that the learning agent exists in an environment described by some set of possible states $S = \{s_1, s_2, ..., s_n\}$. It can perform any of the possible actions in $A = \{a_1, a_2, ..., a_n\}$. The interaction between the agent and the environment at each instant consists of the following sequence:

- The agent senses the state $s_t \in S$.
- Based on $s_t$, the agent performs an action $a_t \in A$.
- As a result, the environment makes a transition to the new state $s_{t+1} = s' \in S$ according to probability $P_{ss'}(a_t)$.
- The agent receives a real-valued reward (payoff) $r_t = r(s_t, a_t)$ that indicates the immediate value of this state-action transition.

**Fig. 1.** Agent-environment interaction.

The task of the agent is to learn a policy, $\pi : S \to A$, for selecting its next action $a_t = \pi(s_t)$ based only on the current state $s_t$. For a policy $\pi$, the *Q*-value $Q^{\pi}(s,a)$ (or state-action value) is the expected discounted cost for executing action *a* at state *s* and then following policy $\pi$ thereafter. The optimal policy $\pi^*(s)$ is the policy that maximizes the total expected discounted rewards $r_t = r(s_t, a_t)$ received over an infinite time. The Q-learning process tries to find $Q^*(s,a) = Q^{\pi^*}(s,a)$ in a recursive manner using available information $(s_t, a_t, s_t', r_t)$ where $s_t$ and $s_t'(= s_{t+1})$ are the states at time *t* and *t*+1 respectively; and $r_t$ is the immediate cost due to $a_t$.

The Q-learning rule is

$$Q_{t+1}(s,a) = \begin{cases} Q_t(s,a) + \alpha_t \Delta Q_t(s,a), & if\ s = s_t and\ a = a_t \\ Q_t(s,a), & otherwise \end{cases} \tag{1}$$

Where

$$\Delta Q_t(s,a) = \left\{ r_t + \gamma \max_b [Q_t(s'_t, b)] \right\} - Q_t(s,a) \tag{2}$$

and $0 \le \gamma \le 1$ is a discount factor and $\alpha_t = \dfrac{1}{1 + visit_t(s_t, a_t)}$ is the learning rate, where $visit_t(s_t, a_t)$ is the total number of times this state-action pair has been visited.

It has been shown [14] that if the *Q*-value of each admissible (*s*,*a*) pair is visited infinitely often, and if the learning rate is decreased to zero in a suitable way, then as $t \to \infty$, $Q_t(x,a)$ converges to $Q^*(x,a)$ with probability 1. The optimal policy $\pi^*(s)$ is the one with the maximum Q-value: $\pi^*(s) = \arg\max_{a \in A(s)} Q^*(s,a)$.

### 2.2 NDP Formulation

This section develops the dynamic programming formulation suitable for the CAC problem in a multimedia cellular network. We consider a fixed channel assignment (FCA) system with *N* available channels in each cell. However, the proposed method can always be easily extended to a dynamic assignment (DCA) scenario.

Let us focus on a given cell and consider for simplicity two classes of traffic C1 and C2. However, the analysis and ideas that follow are easily extendable to cases with several classes of traffic. We assume that $C_1$ calls require just one channel while $C_2$ calls need two channels. Fig. 2 shows for a cell handoffs of the classes of traffic as they come from other neighboring cells.

**Fig. 2.** New and Handoff calls.

This cellular system can be considered as a discrete event system. The major events that may occur in a cell include arrivals and departures of new and handoff calls. These events are modeled as stochastic variables with appropriate probability distributions. In particular, we consider that new call arrivals in a cell obey a Poisson distribution. We also reasonably suppose that handoff traffic is of Poisson type. Call holding time is assumed to be exponentially distributed.

Calls arrive and leave over time and the network can choose to accept or reject connection requests. In return, the network collects revenue (payoff) from these customers. The network operator wants to find a CAC policy that maximizes the long-term revenue/utility and reduces call-blocking probabilities for handoff calls. We set the experimental parameters as shown in Table 1 and 3. We identify system states *s*, actions *a,* and the associated rewards *r* as follows:

1) **States:** At time *t*, the system is in a particular configuration *x* defined by the number of each type of ongoing calls. An event *e* can occur at random times, where *e* indicates either a new or handoff call arrival or a call departure. The departure event is due to a safe termination of a call or a call handoff to a neighboring cell. The configuration *x* and the event *e* together determine the state of the system, *s=(x,e)*. So, we define the state *s=(x,e)* as:

$x=(x_1, x_2)$ where $x_1$ and $x_2$ are the number of calls of each class of traffic ($C_1$ and $C_2$ respectively) in the cell. We notice that as memoryless distributions have been considered, it is not necessary to distinguish new calls and handoff calls in a cell once they are accepted.

We do not take into account the states associated with a call departure for all classes of traffic. The reason for this simplification is that a call departure is not a decision point for the admission controller, and therefore no action needs to be taken.

$e=\{1,2,3,4\}$ where

$$e=\begin{cases}1 & arrival\ of\ a\ new\ call\ of\ class\ C_2\\ 2 & arrival\ of\ a\ new\ call\ of\ class\ C_1\\ 3 & arrival\ of\ a\ handoff\ call\ of\ class\ C_2\\ 4 & arrival\ of\ a\ handoff\ call\ of\ class\ C_1\end{cases}$$

2) **Actions:** Applying an action consists of either to accept or reject the incoming call. So, the possible actions are defined as $A=\{1,0\}$ where

$$a_i = \begin{cases} 1 & accept \\ 0 & reject \end{cases}$$

3) **Rewards:** The reward $r(s,a)$ assesses the immediate payoff incurred due to the acceptance of a call in state $s$. We set the reward parameters, as shown in Table 1, for each class of traffic. To prioritize handoff calls, larger reward values have been chosen for them. We discuss the choice of these reward parameters in section 5.4. When a call of class $i$ arrives ( $e = e_i \in \{1,2,3,4\}$), the reward parameter will be equal to zero when the action is to reject the call ($a=0$) and to $\eta_i$ when the action is to accept it ($a=1$).

Where $$r(s,a) = \begin{cases} \eta_i & if\ a = 1\ and\ e = e_i \\ 0 & otherwise \end{cases}$$

**Table 1.** Immediate Rewards.

| $\eta_1$ | $\eta_2$ | $\eta_3$ | $\eta_4$ |
|---|---|---|---|
| 1 | 5 | 10 | 50 |

To summarize, we choose the state descriptor to be $s = ((x_1, x_2), e)$ , where $x_i$ is the number of class $C_i$ calls in progress, and $e \in \{1,2,3,4\}$ stands for a new or handoff call arrival. When an event occurs, the agent has to choose a feasible action for that event. The action set being $A(s)$={0=reject, 1=accept} upon a call arrival. Call terminations are not decision points so no action needs to be taken. Given $s$, the agent has to determine a call admission policy which maximizes the long-term average revenue over an infinite horizon. The system constitutes an SMDP with a finite state space $S = \{(x, e)\}$ and a finite action space $A$={0,1}.

## 3 Algorithm Implementation

After the specification of the states, actions, and rewards, let us describe the two online implementations of the Q-learning algorithm for solving the CAC problem that we called TQ-CAC and NQ-CAC. These two implementations differ in the way of representing Q-values. TQ-CAC uses a lookup table while NQ-CAC uses a function approximator using a multi-layer neural network. The section below details these issues.

### 3.1 Q-values representation

To represent and store the Q-values, we use two different approaches: a lookup table and a neural network. The differences between these two approaches is shown in Table 2 and are expressed in terms of incurred computational complexities, and storage requirements.

**Table 2.** Look-up table vs. Neural networks.

| | Number of operations | Memory units |
|---|---|---|
| Table | 0 | 676×5=3380 |
| Neural Network | 4×10 + 10×4 + 10 + 10 = 100 | 4×10 + 10 = 50 |

*A. Lookup table*

The lookup table (*cf.* Fig. 3 (a)) is the most straightforward method. The first columns (1 to 3) represent system state $s = ((x_1, x_2), e)$. Columns 4 and 5 represent the acceptance and rejection Q-values respectively. It has the advantage of being both computationally efficient and completely consistent with the structural assumption we have made in order to prove the convergence of the Q-learning scheme [14].

In the case of our two classes of traffic, no multiplication or addition operation is needed as Table 2 shows. But to obtain the Q-values, it is just a matter of indexing which can be very slow since the table-based learning algorithm requires a larger amount of memory units[2] (676×5=3380). Thus, when the input space consisting of state-action pairs is large or the input variables are continuous, the use of lookup tables can be prohibitive because it may imply huge memory requirement. Fortunately, this could significantly reduced through the use of some function approximators such as neural networks or regression trees [12].

*B. Neural Network*

Artificial neural networks (ANN) provide a generic practical method for learning real, discrete, and vector-valued functions from examples. ANN learning is robust to errors in the training data and has been successfully applied to problems such as visual scenes interpretation, speech recognition, and strategies for robot control. The study of ANNs has been inspired in part by the observation that biological learning systems are built out of very complicated webs of interconnected neurons. In rough analogy, artificial neural networks are built out of densely interconnected sets of simple units, where each unit takes a number of real-valued inputs and produces a single real-valued output which may also serve as an input to other units. The reader should refer to [12] for further details.

The multilayer neural network of Fig. 3 (b) has 4 inputs representing the system state $s = ((x_1, x_2), e)$, and the action *a*. It also shows 10 hidden units for internal calculus, and one output unit representing the corresponding Q-value. The computational complexity depends essentially on the network size. In our case, and as shown in

Table 2, 100 operations (multiplications or additions) are required to compute acceptance and rejection Q-values. Fortunately, in terms of storage requirements, the neural network-based Q–learning approach needs a much lower number of memory units to store the network weights. In Fig. 3 it needs only 50 Memory cells.

**Fig. 3.** Q-values representation using (a) Lookup table (b) Neural network.

### 3.2 Implementation

The flowchart of Fig. 4 summarizes the procedures involved in the TQ-CAC and NQ-CAC algorithms. We set the initial Q-values to random values. Since the only interesting states in which decisions need to be made are those associated with call arrivals, we avoid updating of Q-values at departure states. This significantly reduces computation and storage requirements for Q-values.

When there is a call arrival (new or handoff call), the algorithm first verifies whether its acceptance incurs any QoS violation and in which case it is rejected. Otherwise, an action $a$ is chosen where $a$ verifies:

$$a = \arg\max_{a \in A(s)} Q^*(s, a) \tag{3}$$

Where $A(s)$={1=accept, 0=reject}.

In particular, (3) implies the following procedures. When a call arrives, the Q-value of accepting it and the Q-value of rejecting it are determined either from the lookup table (TQ-CAC) or the neural network (NQ-CAC). If rejection has the higher value, the call is dropped. Otherwise, acceptance is higher and the call is accepted.

In both cases, and to learn the optimal Q-values $Q^*(s,a)$, our value function is updated at each transition from state $s$ to $s'$ under action $a$ for the two algorithms as follows:

1. TQ-CAC: (1) is used to update the appropriate Q-value in the lookup table.
2. NQ-CAC: When the Q-value is stored in a neural network, a second learning procedure using the *back-propagation (BP)* algorithm [12] is necessary to learn the weight parameters associated with the network. In this case, $\Delta Q$ defined in (2) serves as an error signal which is propagated back in the neural network as shown in Fig. 3 (b).

[2] The real state space (12×24×4×2 = 2304 states) is reduced to 676 states by eliminating impossible state combinations.

**Fig. 4.** TQ-CAC and NQ-CAC algorithms.

### 3.3 Exploration

Basically, the convergence theorem of Q-learning requires that all state-action pairs (*s*,*a*) are tried infinitely. In order to overcome slow convergence in front of multiple feasible actions during training period, the control action is selected such that it leads to the least visited configuration with probability $\varepsilon$ instead of being selected using (3). That is:

$$a = \arg\min_{a \in A(s)} visits(s,a) \tag{4}$$

This heuristic, named $\varepsilon - directed$, significantly speeds up the convergence of the Q-value function. Q-values are first learned during a sufficiently long period using this heuristic through an offline learning scheme and with the parameters given in Table 3. These values are then used to set up the initial Q-values in our online algorithms.

## 4 Analysis of the Guard Channel Scheme

In this section, we propose to compare our solution with the Guard Channel mechanism. The guard channel scheme is a priority based resource access scheme that allows new and handoff calls to share cell capacity as well as maintaining a certain resource access priority to handoff calls. This is accomplished by fixing a larger resource capacity limit to handoff calls. In this way, the guard channel is consequently the difference between the two capacity limits. The parameters of this mechanism have to be computed off-line for each period of time.

The problem discussed in this part is the following: as different traffic classes are considered, how many channels may be reserved to handoff/new calls of the different classes?

The number of guard channels may be determined for each traffic load and for each traffic class. In each period, *K* classes characterize the traffic. The arriving new and handoff calls in each cell obey a Poisson distribution each with a specific rate $\lambda_j, 1 \le j \le K$. New and handoff calls will be processed by a cell during a time period exponentially distributed with parameter $\mu_j$.

The guard channel mechanism will be characterized by a vector *s* corresponding to the different thresholds, $s = (s_1, s_2, \cdots, s_K)$[3]. These thresholds depend on the traffic type ($C_1$/$C_2$) and traffic nature (new/handoff call). Consequently there will be 4 classes of traffic (new/handoff calls of $C_1$ or $C_2$ types).

[3] $s_j$ is the number of channels for classes 1, 2, .., *j*.

The reward function *F(T)* may be described as follows. Let $\eta_i$ be the reward generated by the acceptance of a class *i* call and $A_i(T)$ the number of accepted class *i* calls between 0 and *T*. This leads to:

$$F(T) = \frac{1}{T}\sum_{i=1}^{K} \eta_i A_i(T) \qquad (5)$$

*F(T)* is consequently the time average of the reward generated by the system.

In the classical guard channel conditions, the system can be modeled by a simple multiclass M/M/N/N queue with trunk reservation. The system is consequently ergodic and we simply obtain,

$$\lim_{T\to\infty} \frac{A_i(T)}{T} = \Lambda_i^s = \lambda_i \left(1 - P_{b,i}^s\right)$$

where $\Lambda_i^s$ is the accepted flow of class *i* at steady state and $P_{b,i}^s$ the blocking probability of class *i* traffic given the threshold vector *s*.

Consequently, the reward generated by such a system is equal to:

$$F^s = \lim_{T\to\infty} F(T) = \sum_{i=1}^{K} \eta_i \Lambda_i^s \qquad (6)$$

The aim is consequently to derive the steady state blocking probabilities in a given configuration *s* and the best vector $s^*$. Let $\Omega_s$ be the state space when considering a thereshold vector *s*.

Under the considered traffic conditions, the vector $\{N(t) = (N_1(t), N_2(t), \cdots, N_K(t)), t \in IR\}$, where $N_i(t)$ is the number of calls of class *i* at time *t*, is a Markov process. This process is also ergodic but the steady state probability is not of product form (the admission policy does not lead to a coordinate convex set [15]). In the case where a large number of classes is considered, this problem should be solved using an approximated product form solution [15].

In the present paper, an exact numerical solution has been derived. Let $\pi_n$ be the steady state probability of $n$. Let *b* be the vector of the resource number needed by the different classes of traffic: $b = (b_1, b_2, \cdots, b_K)$. The blocking probabilities may be expressed as follows:

$$P_{b,i}^s = \sum_{n\in\Omega_s} \pi_n \cdot 1_{\{n\bullet b + b_i > s_i\}} \text{ where } n \bullet b = \sum_{j=1}^{K} n_j \times b_j \qquad (7)$$

To determine the optimal vector $s^*$, the different classes were numbered as follows: $\eta_1 \le \eta_2 \le \cdots \le \eta_K$. For each value of s, the different blocking probabilities can be derived using the previous method. Using (6), $F^s$ is

determined. All the configurations $s$ for which $s_1 \le s_2 \le \cdots \le s_K = N$, where $N$ is the number of channels in the cell, were investigated. In a more general case, more refined solutions should be implemented.

## 5 Simulation

In order to evaluate the benefits of our call admission control algorithms, we simulate a mobile communication system using a discrete event simulation. As stated before, we consider a fixed channel assignment (FCA) system with $N$=24 channels in each cell. The performance of the algorithms has been evaluated by measuring the total rewards, total lost rewards, and handoff blocking probabilities. The learning rate is chosen to be $\gamma = 0.5$ and the exploration probability[4] $\varepsilon = 1$. The major procedures involved in the simulations are summarized in Fig. 4.

A set of simulations was carried out, including the cases of constant traffic load for all traffic classes, varying traffic load, and time-varying traffic load. This set of experiments was applied to situations where a lookup table and a multilayer neural network were used to represent the Q-values. The experimental results are shown in Fig. 5 through Fig. 14. The results show that reinforcement learning is a good solution for the call admission control problem. The proposed algorithms are considerably powerful compared to the greedy policy and to the guard channel scheme presented in section 4. In all cases the lost reward due to the rejection of customers and the blocking probability of handoff calls are significantly reduced. The total rewards due to the acceptance of customers are also increased.

### 5.1 Constant Traffic Load

Our first set of experiments involved a constant traffic load for two classes of traffic $C_1$ and $C_2$. The parameters used in the simulation are given in Table 1 and 3. The call holding time within a cell is exponentially distributed with parameter $\mu$ ($1/\mu$ = 40s).

**Table 3.** Experimental Parameters.

| | Source Type | |
|---|---|---|
| | $C_1$ | $C_2$ |
| Number of channels | 1 | 2 |
| Call holding time ($1/\mu$) | 40 s | 40 s |
| Call arrival rate ($\lambda$) | $\lambda_1 = 180$ *calls / hour* | $\lambda_2 = \lambda_1 / 2 = 90$ *calls / hour* |

Fig. 5(a) shows the total rewards of using learning with the table structure (TQ-CAC) and with a neural network structure (NQ-CAC) computed each ten minutes over one simulated hour. We can see in Fig. 5(a) that

the total rewards due to the acceptance of new or handoff calls of the two classes of traffic ($C_1$ or $C_2$) in the cell are not much higher compared to those of the greedy policy (the gains are about 10%). But, Fig. 5(b) shows that the total lost rewards due to the rejection of new calls or the failure of handoff calls were reduced significantly and the gains using NQ-CAC can reach 71%. From the results shown in Fig. 6, we see that the handoff blocking probability decreases significantly using Q-learning compared to the greedy policy. This metric is given by

$$P_{HO} = \frac{number\ of\ handoff\ calls\ blocked\ in\ the\ system}{number\ of\ handoff\ call\ arrivals\ to\ the\ system} \tag{8}$$

**Fig. 5.** Total rewards per hour (b) Total Lost rewards per hour.

**Fig. 6.** Handoff blocking probability.

We also compare the handoff blocking probability of $C_1$ traffic vs. $C_2$ traffic using TQ-CAC algorithm (Fig. 7(a)) and NQ-CAC (Fig. 7(b)). These metrics are given, for each class of traffic $C_i$, by

$$P_{HO(C_i)} = \frac{number\ of\ handoff\ calls\ of\ type\ C_i\ blocked\ in\ the\ system}{number\ of\ handoff\ call\ arrivals\ of\ type\ C_i\ to\ the\ system} \tag{9}$$

Since $C_1$ calls have priority over $C_2$ calls[5], we notice that the handoff blocking probability of $C_1$ $P_{HO(C1)}$ is less than the handoff blocking probability of $C_2$ $P_{HO(C2)}$.

**Fig. 7.** (a) Handoff blocking probability of $C_1$ vs. $C_2$ using (a) TQ-CAC (b) NQ-CAC.

### 5.2 Varying Traffic Load

In this case we used the policy learned in the previous case (*constant traffic load*) but with six different traffic load conditions (for both classes $C_1$ and $C_2$). It is also shown in Fig. 8 to Fig. 11 that the proposed algorithms result in significant gains compared with alternative heuristics for all the considered traffic loads and especially when the traffic load is heavy. We also notice that when the traffic load is low, the performance advantage of TQ-CAC and NQ-CAC becomes negligible. This can be easily explained by the fact that when the traffic load is low, there are enough channels for all the calls and so all the calls are accepted. This explains why, in Fig. 9, the blocking probability is nearly equal to *0* when the traffic load is set to *120 calls/h* for $C_1$ and *60 calls/h* for $C_2$. It is also clear from the Fig. 10 that the handoff blocking probabilities of $C_1$ calculated using *(9)* are less than the handoff blocking probability of $C_2$ for all the considered traffic loads, and that gains can reach a maximal value of 85%.

---

[4] The exploration mode was used only to train schemes in the training period.
[5] The rewards associated to $C_1$ calls are higher than those of $C_2$ calls (cf. Table 1).

Two experiments were carried out. The first one compares the two different Q-learning based call admission control technique (TQ-CAC and NQ-CAC) implementations. The second compares these two implementations with other call admission schemes.

*A. TQ-CAC vs. NQ-CAC*

It is shown that TQ-CAC and NQ-CAC are equivalent, and that TQ-CAC is even better (cf. Fig. 8(b)). This is due to the fact that NQ-CAC uses a function approximator (neural network) to represent the Q-values which needs more time to converge. The computational complexities and storage requirements are quite different for these two algorithms. These issues are discussed in details in Section 3.1.

**Fig. 8.** (a) Total rewards per hour (b) Total Lost rewards per hour.

**Fig. 9.** Handoff blocking probability with six different traffic loads.

**Fig. 10.** (a) Handoff blocking probability of $C_1$ vs. $C_2$ using (a) TQ-CAC (b) NQ-CAC.

These results illustrate clearly that TQ-CAC and NQ-CAC have the potential to significantly improve the performance of the system over a large range of network loads. It is interesting to notice that neither the table nor the neural networks were retrained for the other traffic loads. This shows that the system possesses some generalization capabilities.

*B. Q-learning vs. Guard channel schemes*

We also compared our results to those obtained with: (1) the guard channel with fixed thresholds – these thresholds were calculated for the constant traffic load given in Table 3; (2) the guard channel with optimized thresholds - (optimal vector $\mathbf{s}^*$ that maximizes $F^s$) are derived for each input traffic load; (3) Analytical greedy – computed through an exact analytical model (M/M/N/N multiclass queue). From the results shown in Fig. 11(a), we see that the gains due to RL are about 16% and are about 35% in Fig. 11(b).

**Fig. 11.** (a) Total rewards per hour (b) Total Lost rewards per hour.

We notice in Fig. 11, that the optimal threshold method leads to better performance results than Q-learning. But in this method we must compute the optimal values for each traffic in an off-line manner. In contrast, and as mentioned before, the Q-learning based call admission control system owns some generalization and adaptation capabilities, and therefore the optimal values are not recomputed for each traffic load.

### 5.3 Time-Varying Traffic Load

The traffic load in a cellular system typically varies by time. In this case, we always use the policy learnt in the first case (*constant traffic load*) and we use, as in [4], the pattern given in Fig. 12 concerning arrivals during a typical 24-h business day. Peak hours generally happen to be at 11:00 a.m., and 4:00 p.m. Fig. 13 gives simulation results under the assumption that the two traffic classes followed the same time-varying pattern as given in Fig. 12. The maximal traffic load is set to be 180 calls/h for class $C_1$ and 90 calls/h for class $C_2$. Blocking probabilities were calculated on an hour-by-hour basis. The improvements of the proposed reinforcement learning algorithms over the greedy policy are apparent specially when the traffic is heavy (at 11:00 a.m. and 4:00 p.m.). As in the last experience, the optimal threshold method leads to better performance results than Q-learning. However, we notice that NQ-CAC is more powerful than TQ-CAC in this case. This implies that the neural network-based Q–learning approach (NQ-CAC) has a real potential to adapt to a time-varying traffic load.

**Fig. 12.** A traffic pattern of a typical business day.

**Fig. 13.** Performance with time-varying traffic load.

### 5.4 Reward varying

Q-learning is one form of reinforcement learning in which the agent learns an evaluation function over states and actions. In particular, the evaluation function $Q(s,a)$ is defined as the maximum expected discounted cumulative reward the agent can achieve by applying action $a$ to state $s$. Hence, the choice of the reward $r$ associated to each action $a$ is imperative. We do not consider, in this work, the set of rewards as representing the real cost of serving each class of traffic. The rewards in Table 1 represent a certain high level preference between these classes of traffic. We give large values for calls of $C_1$ traffic since we prefer the calls which consume less (only one channel). We set in Table 1 the reward of handoff calls 10 times higher than that of newly arrived. This ratio is a critical parameter for comparison of the performance of the various algorithms. In Table 4, we can see the impact of this ratio and show how would be the performance of the various algorithms if it were equal to 3, 5, and 10.

**Table 4.** Impact of the reward values.

| Reward parameters | | Ratio | HO blocking probability gains of $C_1$ vs. $C_2$ |
|---|---|---|---|
| $\eta_1 = 2$ | $\eta_3 = 6$ | 3 | 70% |
| $\eta_2 = 1$ | $\eta_4 = 3$ | | |
| $\eta_1 = 2$ | $\eta_3 = 10$ | 5 | 75% |
| $\eta_2 = 1$ | $\eta_4 = 5$ | | |
| $\eta_1 = 2$ | $\eta_3 = 20$ | 10 | 83% |
| $\eta_2 = 1$ | $\eta_4 = 10$ | | |

### 5.5 On-line learning

Finally, we observed the on-line performance of Q-learning when both learning and admission control operations were carried out simultaneously. Hence, we used the policy learned in the first case (*constant traffic load*), and continued the learning process. We assumed that call arrival rates typically vary in time , and are distributed as in the pattern given in Fig. 12 concerning arrivals during a typical 24-h business day. Fig. 14 shows one of the results where the handoff blocking probability was computed over two days (48 hours). Some improvement due to on-line learning can be seen in this figure in the sense that the accumulated blocking probabilities during the second day were generally lower than those during the first day.

**Fig. 14.** Online performance of Q-learning (TQ-CAC).

## 6 Conclusion

In this paper, we presented a new approach to solve the problem of call admission control in a cellular network supporting multiple classes of traffic each with a different capacity requirement. We formulate the problem as an average reward dynamic programming problem (SMDP), but with a very large state space. The rapid growth in the number of states (caused, for example, by increasing the number of traffic classes $C_i$, the number of channels within a cell $N$, etc.) significantly increases the computational complexity. This led to the development of other methods that are different from traditional SMDP methods which are typically computationally infeasible for large-scale problems. The optimal solution is obtained by using a self-learning scheme based on the Q-Learning algorithm. We used TQ-CAC and NQ-CAC which are two different implementations of it. These two CAC algorithms were used to train the admission controller in each cell to make admission decisions for new and handoff calls and were experimentally evaluated. The benefits gained by using TQ-CAC and NQ-CAC are of different folds. First, the learning approach provides a realistic and simple way to obtain an approximate optimal solution where the optimal solution can be very difficult to find using traditional methods. Second, since the proposed schemes perform in a real-time environments, it is possible to

carry out an online learning while still performing the real admission control. Compared to other schemes like the guard channel, the system offers some generalization capability since admission policies were not relearned for each traffic condition. This scheme is much faster to operate since it can be self trained to find an optimal solution that works over a range of load conditions whereas a guard channel scheme is hard to optimize with multiple classes of traffic and requires case by case solutions. So, any unforeseen event due to significant variations in the environment conditions can be considered as a new experience for improving the adaptation and the learning qualities of the system. Finally, the acceptance policy can be determined with very little computational effort. It is also shown that the proposed CAC algorithms result in significant savings compared to other alternative heuristics.

## References


[1] C. H. Yoon, C. K. Un, Performance of personal portable radio telephone systems with and without guard channels, *IEEE Journal on Selected Areas in Communications (JSAC'1993)*, vol. 11, pp. 911-917, August 1993.

[2] S. Tekinay, B. Jabbari, "Handover and channel assignment in mobile cellular networks," *IEEE Communications Magazine*, (29), November 1991.

[3] I. Katzela, M. Naghshineh, " Channel assignement schemes for cellular mobile telecommunications systems," *IEEE Personal Communications Magazine*, June 1996.

[4] J. Nie and S. Haykin, "A Q-Learning based dynamic channel assignment technique for mobile communication systems", *IEEE Transactions on Vehicular Technology*, vol. 48, N°. 5, September 1999.

[5] S. P. Singh and D. P. Bertsekas, "Reinforcement learning for dynamic channel allocation in cellular telephone systems," *Neural Information Processing Systems (NIPS),* vol. 9, in Mozer M. and al. (eds.), MIT Press, pp. 974-980, 1997.

[6] P. Marbach, O. Mihatsch and J. N. Tsitsikils, "Call admission control and routing in integrated services networks using neuro-dynamic programming", *IEEE Journal on Selected Areas in Communications (JSAC'2000)*, vol. 18, N°. 2, pp. 197 –208, February 2000.

[7] H. Tong and T. X. Brown, "Adaptive Call Admission Control under Quality of Service Constraint: a Reinforcement Learning Solution", *IEEE Journal on Selected Areas in Communications (JSAC'2000)*, vol. 18**,** N°. 2, pp. 209-221, February 2000.

[8] R. Ramjee, R. Nagarajan and D. Towsley, "On Optimal Call Admission Control in Cellular Networks", *IEEE INFOCOM*, pp. 43-50, San Francisco, CA, March 1996.

[9] M. Littman and J. Boyan, “Packet routing in dynamically changing networks: A reinforcement learning approach”, Advances in *Neural Information Processing Systems (NIPS)*, vol. 6, pp. 671-678, San Francisco, CA, 1994

[10] N. Banerjee and S. Das, “Fast Determination of QoS-based Multicast Routes in Wireless Networking using Genetic Algorithms”, *ICC 2001*, vol. 8, pp. 2588 -2592, Helsinki, Finland 2001.

[11] Mitra, M. I. Reiman and J. Wang, “Robust dynamic admission control for unified cell and call QoS in statistical multiplexers," *IEEE Journal on Selected Areas in Communications (JSAC’1998)*, vol. 16, no. 5, pp. 692-707, 1998.

[12] T. M. Mitchell, “Machine Learning”, *McGraw-Hill companies, Inc.*, 1997

[13] G. Barto, S. J. Bradtke, and S. P. Songh, “Learning to act using real-time dynamic programming”, *Artificial Intelligence*, vol. 72, pp. 81-138, 1995.

[14] C.J. C. H. Watkins and P. Dayan, “Q-learning”, *Machine Learning*, vol. 8, pp. 279-292, 1992.

[15] COST242, “Multi-rate models for dimensioning and performance evaluation of multiservice networks,”, *M. Ritter, P. Tran-Gia Eds*, June 1994.

[16] P. Marbach, O. Mihatsch, M. Schulte and J. N. Tsitsiklis, “Reinforcement learning for call admission control and routing in integrated service networks," Advances in *Neural Information Processing Systems (NIPS)*, vol. 10, in Jordan M. and al. (eds.), MIT Press, 1998.

[17] P. Marbach, J. N. Tsitsiklis, “A Neuro-Dynamic Approach to Admission Control in ATM Networks: The Single Link Case", *International Conference on Acoustics, Speech, and Signal Processing (ICASSP’97)*, Munich, Germany, April 1997.

[18] H. Tong, Adaptive Admission Control for Broadband Communications, *Ph.D. thesis*, University of Colorado, Boulder, Summer 1999.

[19] T. X Brown, ``Low Power Wireless Communication via Reinforcement Learning ," Advances in *Neural Information Processing Systems (NIPS)*, vol. 12, in S.A. Solla, T.K. Leen and K.-R. Muller (eds.), pp. 893-899, MIT Press, 2000.

[20] S. Senouci, A.-L. Beylot, Guy Pujolle, “A dynamic Q-learning-based call admission control for multimedia cellular networks”, *IEEE International Conference on Mobile and Wireless Communications Networks (MWCN’2001)*, pp. 37-43, Recife, Brazil, August 2001.

[21] S. Senouci, A.-L. Beylot, Guy Pujolle, "Call Admission Control for Multimedia Cellular Networks Using Neuro-Dynamic Programming", *IFIP Networking, NETWORKING'02*, Pisa, Italy, May 2002, Lecture Notes in Computer Science, 2345, pp. 1208-1213, Springer Verlag.

[22] M. Sridharan and G. J. Tesauro, "Multi-agent Q-Learning and Regression Trees for Automated Pricing Decisions," *in Proceedings of Seventeenth International Conference on Machine Learning*, Stanford University, CA, June 29-July 2, 2000.

[23] G. J. Tesauro, "Pricing in Agent Economies Using Neural Networks and Multi-Agent Q-learning," *in Proceedings of Workshop ABS-3: Learning About, From and With other Agents*, Stockholm, August 1999.

[24] A.-L. Beylot, S. Boumerdassi and G. Pujolle, "NACR: A New Adaptive Channel Reservation in Cellular Communication Systems," *Telecommunication Systems*, vol. 17, N°. 1-2, pp. 233-241, Kluwer, 2001.

[25] F. Yu and V. C. M. Leung, "Mobility-Based predictive Call Admission Control and bnadwidth reservation in wireless cellular networks," *IEEE Infocom'01*, Alaska, April 2001.

[26] C. Chao and W. Chen, "Connection admission control for mobile multiple-class personal communications networks," *IEEE Journal on Selected Areas in Communications (JSAC'1997)*, vol. 15, N°. 8, pp. 1618-1626, Oct. 1997.

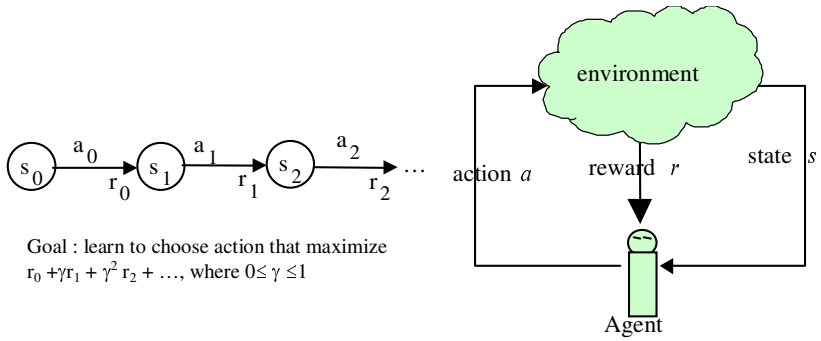



**Fig. 1.** Agent-environment interaction.

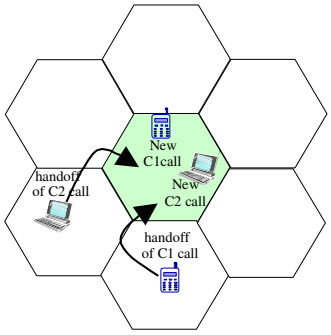
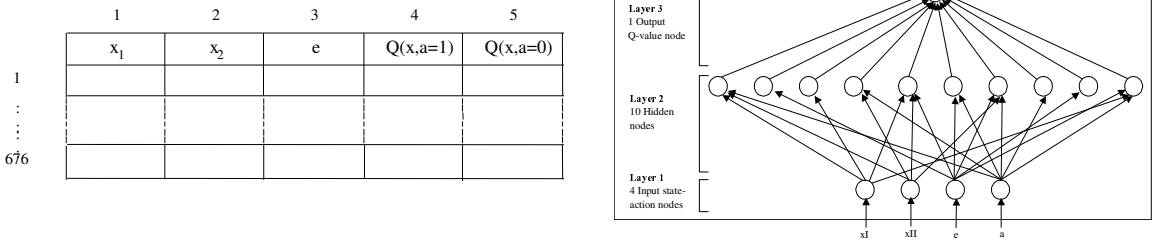



**Fig. 2.** New and Handoff calls.




(a) (b)

**Fig. 3.** Q-values representation using (a) Lookup table (b) Neural network.

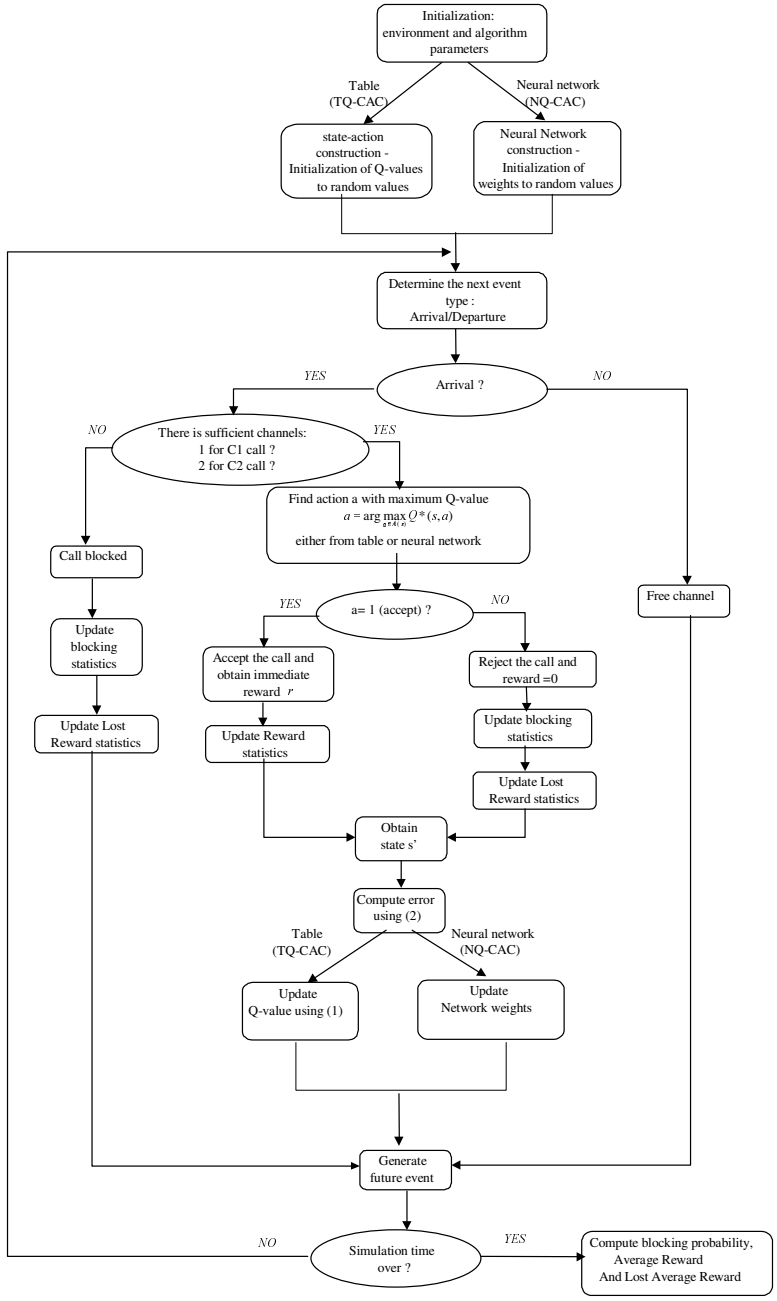
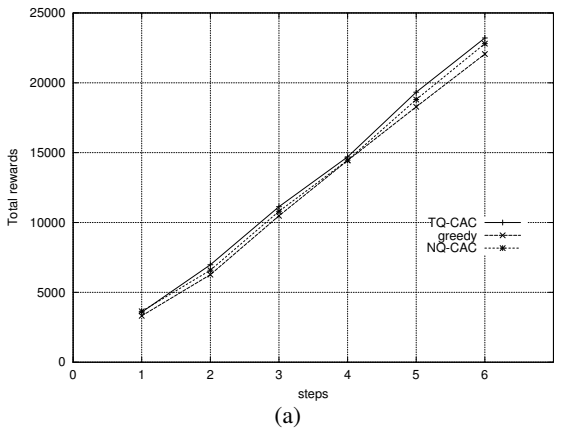
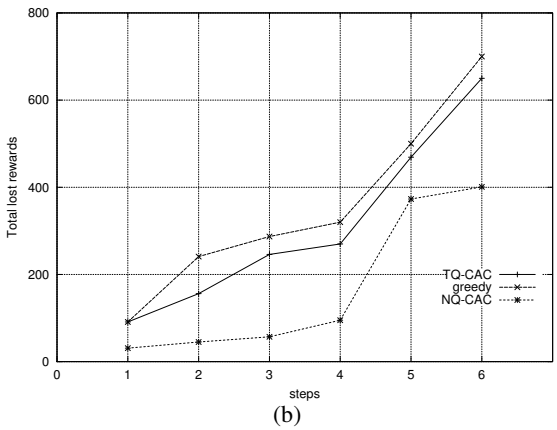



**Fig. 4.** TQ-CAC and NQ-CAC algorithms.




(a)




(b)

**Fig. 5.** Total rewards per hour (b) Total Lost rewards per hour.

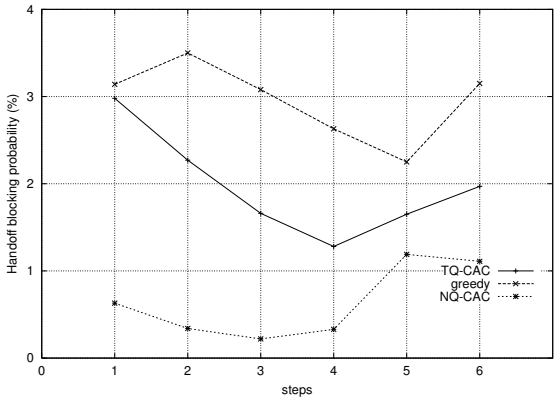
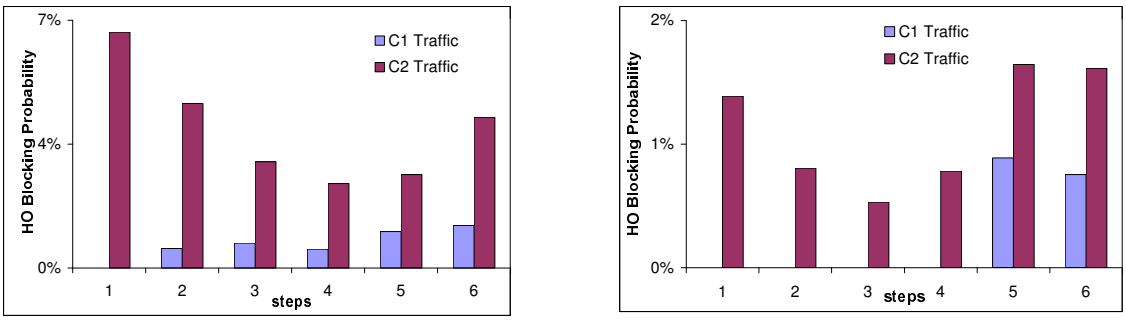
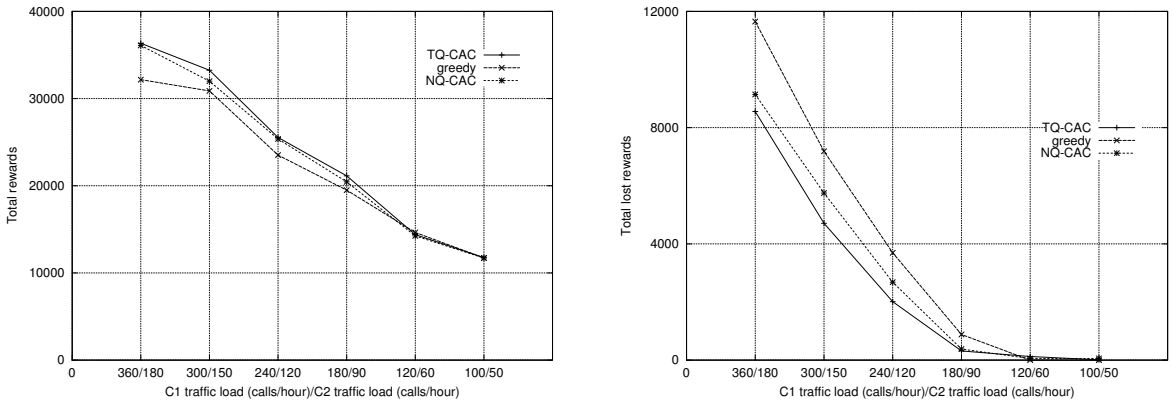



**Fig. 6.** Handoff blocking probability.




(a) (b)

**Fig. 7.** (a) Handoff blocking probability of $C_1$ vs. $C_2$ using (a) TQ-CAC (b) NQ-CAC.




(a) (b)

**Fig. 8.** (a) Total rewards per hour (b) Total Lost rewards per hour.

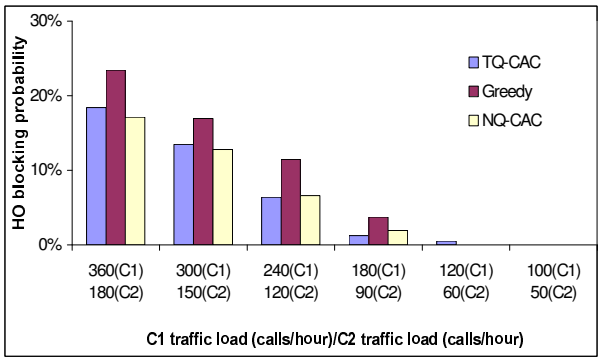
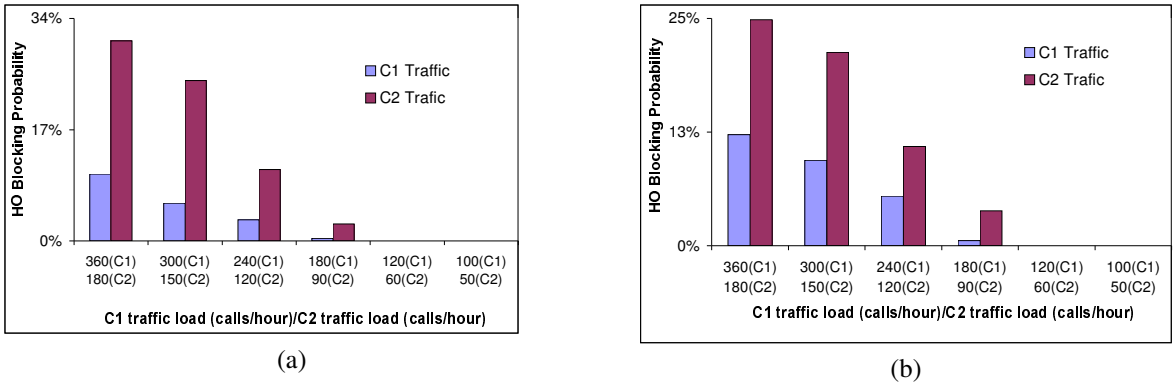
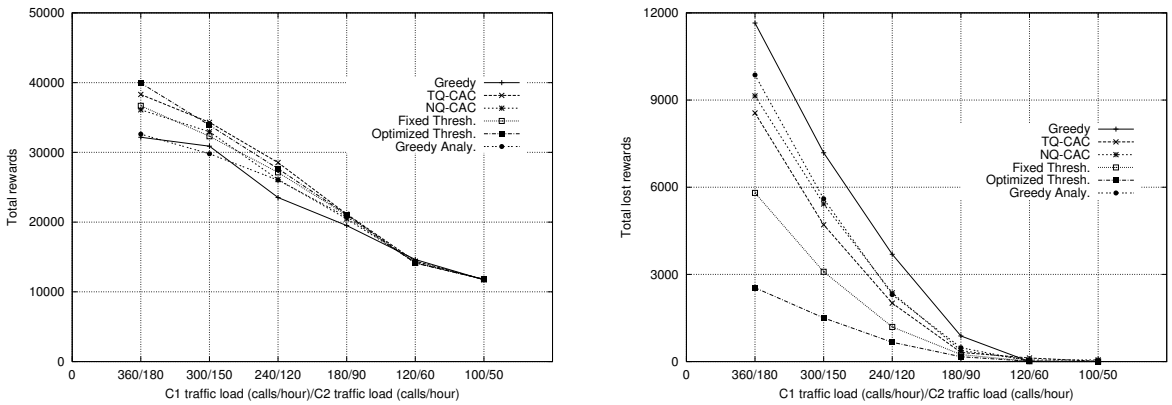


**Fig. 9.** Handoff blocking probability with six different traffic loads.




(a) (b)

**Fig. 10.** (a) Handoff blocking probability of $C_1$ vs. $C_2$ using (a) TQ-CAC (b) NQ-CAC.




(a) (b)

**Fig. 11.** (a) Total rewards per hour (b) Total Lost rewards per hour.

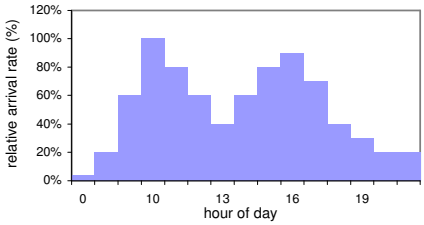



**Fig. 12.** A traffic pattern of a typical business day.

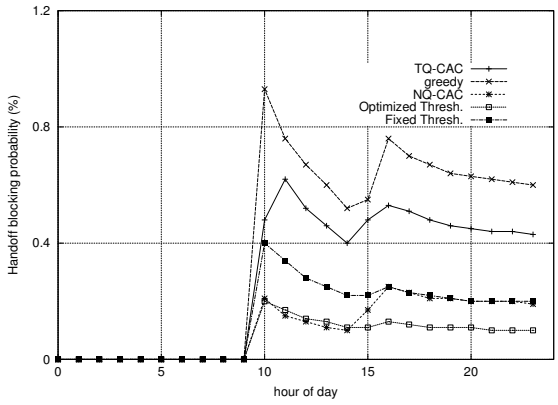


**Fig. 13.** Performance with time-varying traffic load.

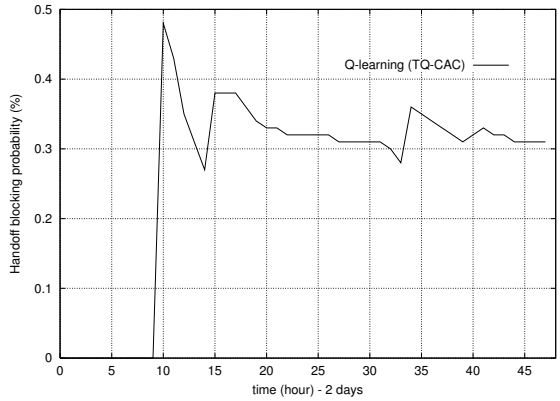



**Fig. 14.** Online performance of Q-learning (TQ-CAC).