A dynamic Q-learning-based call admission control for multimedia cellular networks

Sidi-Mohammed Senouci, André-Luc Beylot, Guy Pujolle

A Dynamic Q-Learning-Based Call Admission Control for Multimedia Cellular Networks

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- **In this paper, we address the call admission control (CAC) problem in a multimedia cellular network that handles several classes of traffic with different resource requirements. We formulate the problem of revenue maximization as a Semi-Markov Decision Process (SMDP) problem. It is too complex to allow for an exact solution for this problem, so, we use a real-time reinforcement learning (RL) algorithm known as Q-Learning algorithm to construct a dynamic call admission control policy. We show that Q-CAC, which is our implementation of the Q-learning algorithm for solving the CAC problem, provides a good solution and is able to earn significantly higher revenues than alternative heuristics. Performance evaluations show that our policy compared to other policies improves the Quality of Service (QoS) and reduces call blocking probabilities for handoff calls in spite of variations in the environment conditions.**

Keywords- **Reinforcement Learning, Q-Learning, dynamic programming, call admission control, multimedia cellular networks.**

I. INTRODUCTION

Recently, the demand for wireless communications which will provide reliable voice and data communications, anytime and anywhere has massively grown. The service area in these networks is partitioned into cells. In each cell, a Base Station (BS) manages the allocation of channels[1] to the Mobile Subscriber (MS) enabling the MS to communicate with other MS's or PSTN users. Note that the BS itself is assigned a set of channels and this assignment could be static or dynamic. We primarily assume a static assignment of channels for this paper but the ideas in the paper can easily be extended to dynamic assignment scenarios as well.

As a MS moves from one cell to another, any active call needs to be allocated a channel in the destination cell. This event (handoff) must be transparent to the MS. If the destination cell has no available channel, the call is terminated. The disconnection of ongoing calls is highly undesirable and one of the goals of the network designer is to keep the handoff blocking probability low. In [6] the authors show that the well-known Guard Channel policy, which reserves a set of channels for handoff calls, is optimal for minimizing this entity.

This technique is simple to dimension in a mono-class traffic framework, but the optimization is quite complicated in a multi-class context. In a multi-class context it is sometimes preferable to block a call of a less valuable class and to accept another one of a more valuable class. Furthermore, these techniques ignore completely the experience or knowledge that could be gained during real-time operation of the system.

In this new context, the use of learning techniques [1,8], can lead to good solutions in reasonable times. Instead of relying on a known teacher, the system is designed to learn an optimal assignment policy by directly interacting with the environment. This policy

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

must be able to reduce the blocking probability for handoff calls and, also, able to produce higher revenues.

A number of researchers have recently explored the application of the learning algorithms like *Reinforcement Learning (RL)*, *MultiLayer Perceptron (MLP)* and Genetic algorithms to resource allocation [3], network routing and admission control [4,5] in telecommunications systems.

This paper proposes an alternative approach to solving the call admission control (CAC) in multimedia cellular networks. The optimal CAC policy is obtained through a form of reinforcement learning known as Q-learning [8,9].

We consider a system with two classes of traffic. We associate to each class of traffic a different reward (payoff) representing the cost for serving a customer of that class. These rewards are, also, different according to the call type (new or handoff call). 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 an SMDP and learning is a solution to this problem.

The rest of the paper is organized as follows. After a brief description of the Q-Learning strategy and the formulation of the CAC problem as an a SMDP and giving the RL algorithm (Q-CAC) that solves the SMDP in section 2, we detail the implementation of the proposed RL algorithm in section 3. Performance evaluation and numerical results are exposed in section 4. Finally, section 5 summarizes the main contributions of this work.

II. PROBLEM DESCRIPTION

We propose an alternative approach to solving the call admission control problem. The approach is based on the judgment that the CAC can be regarded as an SMDP, and learning is one of the effective ways to find a solution to this problem. A particular learning paradigm has been adopted known as *reinforcement learning (RL)*. In RL, 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 (costs) obtained from the environment is maximized. There exists a variety of RL algorithms. A particular algorithm that appears to be suitable for the CAC task is called Q-learning. In what follows, we first describe this algorithm briefly as in [2], and then present the details of how the CAC problem can be solved by means of Q-learning.

A. Q-Learning Strategy

Assume that the learner agent exists in an environment described by some set of possible states $S = \{s_1, s_2, ..., s_n\}$. It can perform any of possible actions $A = \{a_1, a_2, ..., a_n\}$. 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 $P_{ss'}(a)$.
- 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.

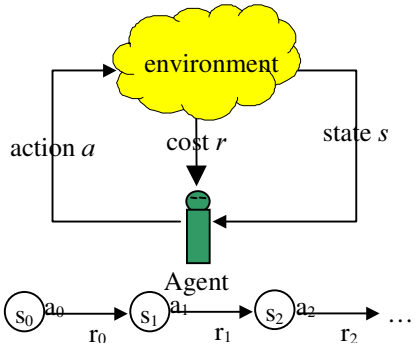


Fig. 1. The Agent-environment interaction.

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 on the current state s_t. The

optimal policy $\pi^*(s)$, is the policy which maximizes the total expected discounted rewards $r_t = r(s_t, a_t)$ received over an infinite time, and defined as

$$V^{\pi}(s) = E\left\{\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \middle| s_0 = s\right\} \quad (1)$$

where E stands for the expectation operator and $0 \le \gamma \le 1$ is a discount factor. $V^{\pi}(s)$ is often called the value function of the state s.
Equation (1) can be rewritten as [2]

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \pi(s)} P_{ss'}(\pi(s)) V^{\pi}(s')$$

where $R(s, \pi(s)) = E\{r(s, \pi(s))\}$ is the mean value of $r(s, \pi(s))$. The optimal policy π^* satisfies Bellman's optimality criterion

$$V^*(s) = V^{\pi^*}(s) = \max_{a \in A}\left[R(s,a) + \gamma \sum_{s' \in S} P_{ss'}(a) V^{\pi}(s')\right] \quad (2)$$

The task of Q-learning is to determine a π^* without knowing $R(s,a)$ and $P_{ss'}(a)$, which makes it well suited for the CAC problem. This is achieved by reformulating (2). For a policy π, define a Q-value (or state-action value) as

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'} P_{ss'}(a) V^{\pi}(s')$$

which is the expected discounted cost for executing action a at state s and then following policy π thereafter.
Let

$$Q^*(s,a) = Q^{\pi^*}(s,a) = R(s,a) + \gamma \sum_{s'} P_{ss'}(a) V^{\pi^*}(s')$$

We then get

$$V^*(s) = \max_{a \in A}[Q^*(s,a)]$$

Thus, the optimal value function V^* that satisfies Bellman's optimality criterion can be obtained from $Q^*(s,a)$ and in turn $Q^*(s,a)$ may be expressed as

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} \left\{P_{ss'}(a)\left[\max_{b \in A} Q^*(s', b)\right]\right\}$$

The Q-learning process tries to find 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 a_t and r_t are the action taken at time t and the immediate cost due to a_t at s_t, respectively.
The Q-learning rule is

$$Q_{t+1}(s,a) = \begin{cases} Q_t(s,a) + \alpha_t \Delta Q_t(s,a), & \text{if } s = s_t \text{ and } a = a_t \\ Q_t(s,a), & \text{otherwise} \end{cases} \quad (3)$$

where

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

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 [9] 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.

B. Learning CAC by Q-Learning

This section develops the SMDP 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. But this can be extended easily to the dynamic assignment (DCA) scenario as well. Let us focus on a given cell.
We also consider two classes of traffic $C1$ and $C2$. But, the ideas in this paper can be extended easily to several classes of traffic as well. Calls of the first class needs only one channel and calls of the second needs two channels. We, of course, consider the handoff calls of these two classes coming from the neighboring cells as represented in *Fig. 2*.

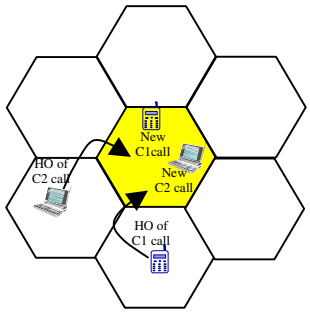


Fig. 2. New and Handoff calls.

This cellular system can be considered as a discrete-time event system. The major events which may occur in a cell include new and handoff call arrivals and call departures. These events are modeled as stochastic variables with appropriate probability distributions. In particular, 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 customers for calls that it accepted or rejected. 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 2.

We identify the system states *s*, the 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. At random times an event *e* can occur, 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)*. We define the state *s=(x,e)* as :
   - $x=(x_I, x_{II})$ where x_I and x_{II} are the number of calls of each class of traffic (*C1* and *C2* respectively) in the cell.

   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 call departure is not a decision point for the admission controller, and therefore no action needs to be taken.
   - $e=\{1,2,3,4\}$

$$e=\begin{cases}1 & arrival\ of\ a\ new\ call\ of\ class\ C1\\ 2 & arrival\ of\ a\ new\ call\ of\ class\ C2\\ 3 & arrival\ of\ a\ HO\ call\ of\ class\ C1\\ 4 & arrival\ of\ a\ HO\ call\ of\ class\ C2\end{cases}$$

2) **actions :** Applying an action is to accept or reject the current 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 acceptation of a call in state *s*. We set the reward parameters, as shown in *table* 2, for each class of traffic. They are different for new calls and for handoff calls. To prioritize handoff calls, larger reward values have been chosen for handoff calls. These values are chosen to be up to 10 times larger compared to the those of the new calls in order to accelerate the algorithm convergence and to have results in a reasonable time. The reward parameter is equal to zero when the action is to reject the call (*a=0*).

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

Table. 1. Immediate rewards.

η_1	η_2	η_3	η_4
5	1	50	10

In summary, we choose the state descriptor to be $s=((x_I,x_{II}),e)$, where x_i is the number of calls of class *i* in progress, and $e \in \{1,2,3,4\}$ stands for a new or handoff call arrival. When an event occurs, the learner has to choose a feasible action for that event. The action set is *A(s)={0=reject, 1=accept}* upon a call arrival. Call terminations are not decision points, so no action needs to be taken.

The learner has to determine a policy for accepting calls given *s*, that maximizes the long-run average revenue, over an infinite horizon. For CAC, the system constitutes an SMDP with a finite state space *S = {(x, e)}* and a finite action space $A=\{0,1\}$.

III. ALGORITHM IMPLEMENTATION

After the specification of the states, actions and costs, let us describe the Q-CAC algorithm which is the online implementation of the Q-learning algorithm for solving the CAC problem.

There exists a variety of approaches to represent and store the Q-values [8]. In this work we used the lookup table. The lookup table is the most straightforward method. It has the advantage of being both computationally efficient and completely consistent with the structure assumption made in proving the convergence of the Q-learning scheme. However, when the input space consisting of state-action pairs is large or the input variables are continuous, using lookup tables can be prohibitive because memory requirement may be huge. In this case, some function approximators such as neural networks [8] may be used in an efficient manner. In [2], the author gives a comparison between these two approaches in term of computational complexities and storage requirements.

We set initial Q-values to zero such that Q-learning started with the greedy policy. We note that the only interesting states in which decisions need to be made are those associated with call arrivals. So, we avoid the updates of Q-values at departure states. This will reduce the amount of computation and storage of Q-values significantly.

We apply a test data set to compare our policy with the greedy policy (the policy that always accepts a new call if the capacity constraint will not be violated by adding the new call). In Q-CAC, when there is a new call arrival, the algorithm first determines if accepting this call will violate QoS. In this case, the call is rejected, else the action is chosen according to

$$a = \arg\max_{a \in A(s)} Q^*(s,a) \qquad (12)$$

where *A(s)={1=accept, 0=reject}*

In particular, (12) implies the following procedures. When a call arrives, the Q-value of accepting the call and the Q-value of rejecting the call are determined. If rejection has the higher value, we drop the call. Otherwise, if acceptance has the higher value, we accept the call.

In these two cases, and to learn the optimal Q-values *Q*(s,a)*, we update our value function at each transition from state *s* to *s'* under action *a* using (3). The flowchart of *Fig. 3* shows the procedures involved in the Q-CAC algorithm.

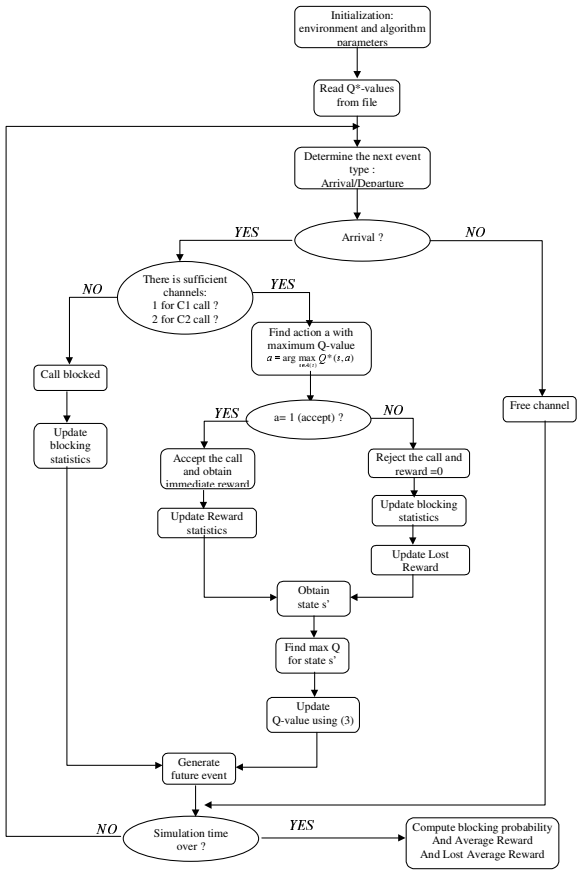


Fig. 3. Q-CAC algorithm.

In order Q-learning to perform well, all the potentially important state-action pairs *(s,a)* have to be explored. Basically, the convergence theorem of Q-learning requires that all state-action pairs *(s, a)* are tried infinitely. To overcome the slow convergence, during the training period, when there are more than one feasible action, the control action is chosen, not according to (12), but the one that leads to the least visited configuration with probability ε. This heuristic, named $\varepsilon - directed$, significantly speeds up the convergence of the value function. The Q-values are first learned with a sufficiently long time period using this

heuristic in an offline learning scheme with the parameters given in *table 2*. These values are used to set up the initial Q-values in our online Q-CAC algorithm. We notice that we can use only the online learning scheme, but this will take a long time before the convergence of the Q-values. The offline learning scheme permits the exploration of all potentially state-action pairs.

IV. SIMULATION

In order to evaluate the benefits of our call admission control algorithm (Q-CAC), 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 algorithm was evaluated on the basis of the total rewards of accepted calls (*Total rewards*), the total rewards of the rejected calls (*Total Lost Rewards*) and by measuring the handoff blocking probability. In simulations, the learning rate is chosen to be $\gamma = 0.5$, and exploration probability $\varepsilon = 1$. The major procedures involved in the simulations are summarized in *Fig. 3* .

A set of simulations were carried out, including the cases of a constant traffic load for all traffic classes, a traffic load varying and a time varying traffic load. The experimental results are given in *Fig. 4* through *Fig. 8*. The main conclusion from these results is that the reinforcement learning seem to be promising for the call admission control problem. Q-CAC leads to significantly better results than the greedy policy. In all cases the lost reward due to rejection of customers and blocking probability of handoff calls are significantly reduced. The total rewards due to acceptance of customers is also significantly increased.

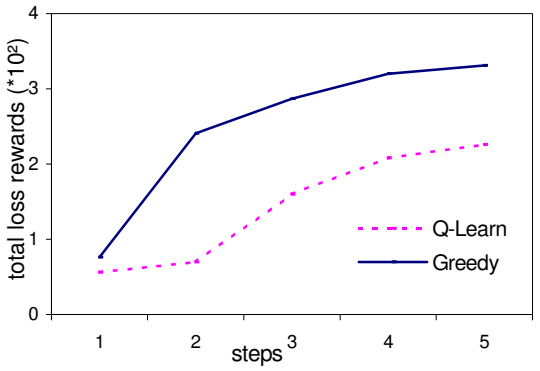
1) Constant Traffic Load

Our first set of experiments involved a constant traffic load for the classes of traffic *C*1 and *C*2. The parameters used in the simulation are given in *table 1* and *2*.

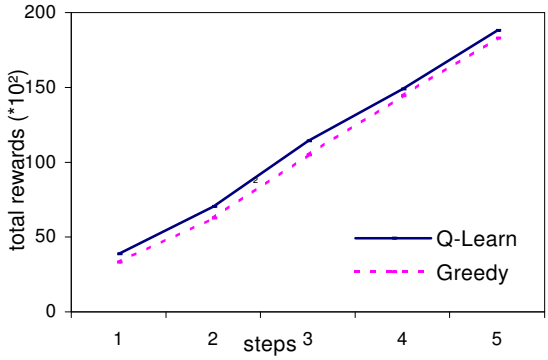
The parameters were chosen as follows. The call duration is assumed to be exponentially distributed with parameter μ_C ($1/\mu_C$ = 120s). The sojourn time of a mobile user within a cell is also supposed to be exponentially distributed with parameter μ_H ($1/\mu_H$ = 60s). Consequently, the call holding time is exponentially distributed with parameter $\mu=\mu_C + \mu_H$ ($1/\mu$ = 40s).

Table 2. Experimental parameters.

Parameter	Source Type	
	C1	C2
Number of channels	1	2
Call holding time	40 s	40 s
Call arrival rate	$\lambda_1 = 180\ calls/h$	$\lambda_2 = \lambda_1/2 = 90\ calls/h$



(a)



(b)

Fig. 4 (a) Total rewards per hour (b) Total Loss rewards per hour

Fig. 4 shows the total rewards calculated each ten minutes over one simulated hour. We can see in *Fig. 4 (a)* that the total rewards due to acceptance of new or handoff calls of the two classes of traffic (*C1* or *C2*) in the cell after one hour is more important compared to the total rewards of the greedy policy. In Fig. 4 (b) we can, clearly, see that the total loss rewards due to rejection of new calls or the failure of handoff calls could be reduced significantly.

2) Traffic Load varying

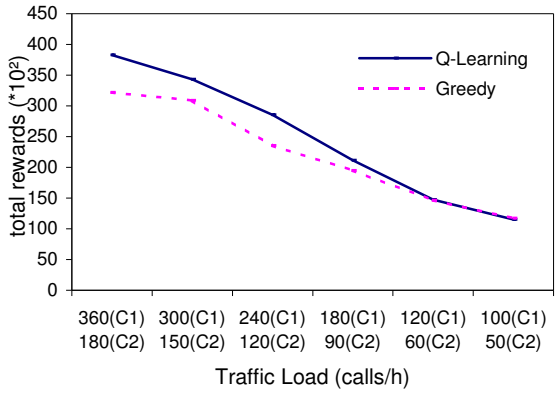
In this case we use the same policy learned in the first case (*constant traffic load*) but with six different traffic load conditions (for both classes C1 and C2) as shown in *table 3*.

Table 3. Experimental parameters

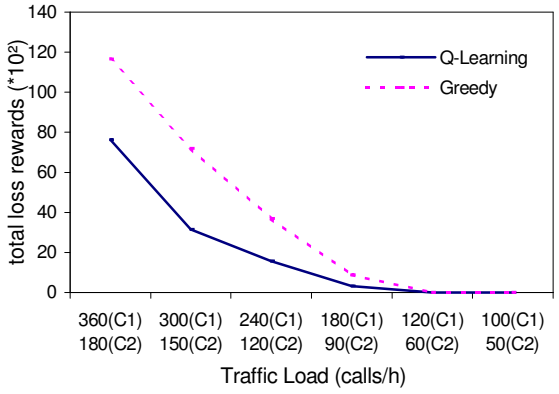
	Sources Type	
	C1	C2
Traffic Load (calls/h)	360	180
	300	150
	240	120
	180	90
	120	60
	90	45

Compared with the greedy policy, Q-CAC policy reduces the blocking probability of the handoff calls as shown in *Fig. 6*. It is also shown in *Fig.5* that the Q-CAC algorithm results in significant gains compared with alternative heuristics in term of total rewards and total loss rewards received after one simulated hour for all the traffic loads considered in *table 3* and especially when the traffic load is heavy .

We can note that when the traffic load is slight, the performance advantage of Q-CAC over the greedy policy become negligible. This can be easily explained by the fact that when the traffic load is slight, there is enough channels for all the calls and so all the calls are accepted. This explains why, in *Fig. 6*, the blocking probability is nearly equal to *0* when the traffic load is set to *120 calls/h* for the class of traffic C1 and *60 calls/h* for the class of traffic C2.



(a)



(b)

Fig. 5 (a) Total rewards per hour (b) Total Loss rewards per hour

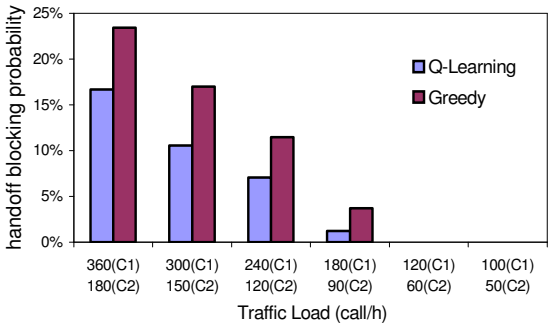


Fig. 6. Handoff blocking with six different traffic loads.

This illustrates clearly that Q-CAC has the potential to significantly improve performance over a broad range of network loads.

3) Time-Varying Traffic Load

The traffic load in a cellular system is typically time varying. In this case, we always use the same policy learned in the first case (*constant traffic load*) and we take, as in [3], the pattern given in *Fig. 7* concerning arrivals during a typical 24-h business day. The peak hours occur at 11:00 a.m. and 4:00 p.m. *Fig. 8* gives the simulation results under the assumption that the two traffic classes followed the same time-varying pattern given in *Fig. 7*. The maximum traffic load is set to be 180 calls/h for class C1 and 90 calls/h for class C2. The blocking probabilities were calculated on an hour-by-hour basis. The improvement of the Q-CAC over the greedy policy is apparent specially when the traffic is heavy (at 11:00 a.m. and 4:00 p.m.).

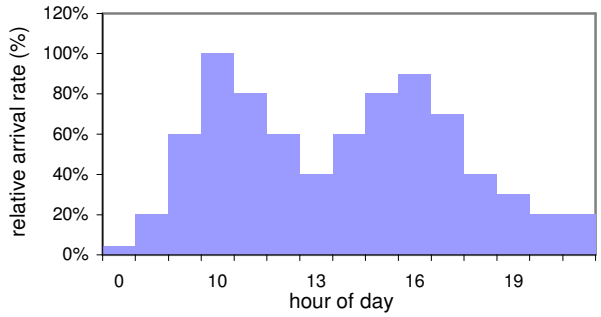


Fig. 7 . A traffic pattern of a typical business day.

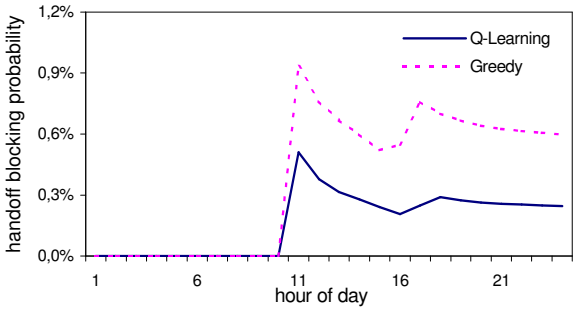


Fig. 8 . Performance with time-varying traffic load.

V. CONCLUSION

In this paper, we presented a new approach to solving the problem of call admission control in a cellular multimedia network. We formulate the problem as an average reward dynamic programming problem (SMDP), but with a very large state space. Traditional SMDP methods are computationally infeasible for such large scale problems. So, the optimal solution is obtained by using a self-learning scheme based on Q-Learning algorithm. The benefits gained by using Q-CAC are as follows. First, the learning approach provides a realistic and simple way to obtain an approximate optimal solution for which an optimal solution can be very difficult to find using traditional methods (SMDP). Second, since the proposed Q-CAC scheme is performed in real-time environment, it is possible to carry out online learning while performing the real admission control. In this way, any unforeseen event occurring due to significant variations in the environment conditions, such as traffic load, can be considered as a new experience that the system could utilize for improving its learning quality. Third, the computational requirements are slow and the acceptation policy can be determined with very little computational effort. It is, also, shown that the Q-CAC algorithm results in significant savings than alternative heuristics. Prospective works deal with comparison with other classical solutions such as Trunk Reservation (Guard Channel) or Virtual Partitioning and with other learning algorithms such as Neural Networks.

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] M. Littman and J. Boyan, "Packet routing in dynamically changing networks: A reinforcement learning approach", *Neural Information Processing Systems (NIPS)*, vol. 6, pp. 671-678, San Francisco, CA, 1993.

[8] T. M. Mitchell, "Machine Learning", *McGraw-Hill companies, Inc.*, 1997.

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