New Channel Assignments in Cellular Networks: A Reinforcement Learning Solution

Sidi-Mohammed Senouci, and 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

Abstract- **The optimization of channel assignment in cellular networks is a very complex optimization problem and it becomes more difficult when the network handles different classes of traffic. The objective is that channel utility be maximized so as to maximize service in a stochastic caller environment. We address in this paper, the dynamic channel assignment (DCA) combined with call admission control (CAC) problem in a multimedia cellular network that handles several classes of traffic with different resource requirements. The problem is naturally formulated as a Semi-Markov Decision Process (SMDP) problem and we use an approach based on reinforcement learning (RL) [neuro-dynamic programming (NDP)] method to solving it. An additional solution to construct a call admission control (CAC) policy in an FCA (Fixed Channel Assignment) system through the same approach (RL) is proposed in this paper. We show that the policies obtained using our algorithms QCAC-FCA and Q-DCA provide a good solution and are able to earn significantly higher revenues than classical solutions. A broad set of experiments illustrates the robustness of our policies that improve the Quality of Service (QoS) and reduce call-blocking probabilities for handoff calls in spite of variations in the traffic conditions.**

1 Introduction

Technological advances and rapid development of handheld wireless terminals have facilitated the rapid growth of wireless communications and mobile computing. Taking ergonomic and economic factors into account, and considering the new trend in the telecommunications industry to provide ubiquitous information access, the population of mobile users will continue to grow at a tremendous rate. The tremendous growth of the wireless/mobile user population, coupled with the bandwidth requirements of multimedia applications, requires efficient reuse of the scarce radio spectrum allocated to wireless/mobile communications.

The total system bandwidth is divided into channels[1], with each channel centered around a frequency and the most important problem is to allocate these channels so as to maximize the service provided to a set of mobile callers. The assignment of this bandwidth fall into two categories: Fixed Channel Allocation (FCA), where each cell has a fixed number of channels, and dynamic channel allocation (DCA) where channels are dynamically assigned to cells. In FCA, the set of channels is partitioned according to some reuse pattern, and the partitions are permanently assigned to cells. When a call arrives in a cell, if any pre-assigned channel is unused; it is assigned, else the call is blocked. Such policies are very simple, however, they do not adapt to changing traffic conditions and user distribution. More efficient are DCA policies, where all channels are placed in a pool and are assigned to new calls as needed such that the carrier-to-interference ratio (CIR) criterion is satisfied. At the cost of higher complexity, DCA schemes provide flexibility and traffic adaptability.

In [1] the authors provide an overview of different channel assignment algorithms and compare them in terms of performance, flexibility, and complexity. One of the best existing dynamic channel allocation strategies we found in the literature belongs to a class of algorithms called exhaustive searching DCA [2,3,4-7]. In these algorithms, a cost (reward) is associated with each available channel. When a new call arrives, the system searches exhaustively for the channel with minimum cost (maximum reward) and then that channel is assigned to the call. Some criteria including maximum availability, maximum interferers, and minimum damage have been used.

In both FCA and DCA systems, when a mobile caller crosses from one cell to another, he needs to be allocated a new channel (one that does not violate the channel reuse[2] constraint) in the destination cell. This event (handoff) must be transparent to the user. If no such channel is available, the call must be dropped/disconnected from the system. One objective of a channel allocation policy is to minimize the number

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

[2] A channel can be associated with many cells as long as the co-channel interference constraint is satisfied.

of calls that are dropped when they are handed off to a busy cell, since dropping existing calls is generally more undesirable than blocking new calls.

This paper proposes an alternative approach to solve two problems in cellular networks. The first one is the channel assignment and call admission control problems in DCA systems. The second one is the call admission control problem in FCA systems. The optimal policies are obtained using a form of reinforcement learning (RL) algorithm known as Q-learning [8]. Reinforcement learning procedures have been established as powerful and practical methods for solving Markov Decision Problems. One of the most significant and actively investigated RL algorithms is Q-learning. It has the nice property that it does not need a model of the environment, and the system is designed to learn an optimal policy by directly interacting with the environment. Our methods learn policies that outperform the most commonly used policies in cellular systems. They are able to reduce the blocking probability for handoff calls and, also, able to generate higher revenues.

We consider a system with two classes of traffic. Our objective is to accept or reject customers for the second problem, and to assign the best available channel to the customer for the first one so as to maximize the expected value of the rewards received over an infinite planning horizon. In such context (multi-class traffic framework) it is sometimes preferable to block a call of a less valuable class and to accept another call of a more valuable class. 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 for this problem.

The remainder of this paper is organized as follows. After a brief description of the Q-Learning strategy and the formulation of the two problems as an SMDP in section 2, we detail the different implementations of Q-learning algorithm that solves these SMDP in section 3. Performance evaluation and numerical results are exposed in section 4. Finally, section 5 summarizes the main contributions of this work.

2 Problem definition

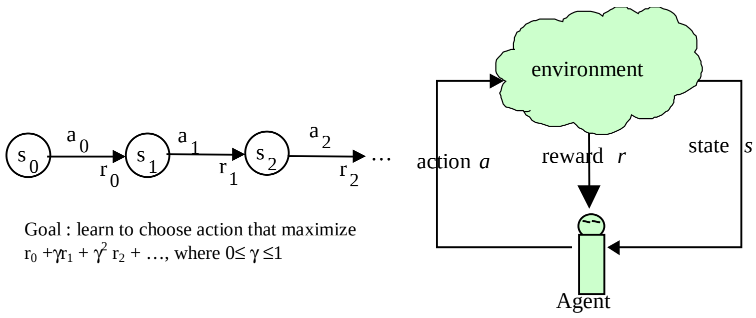
We propose an alternative approach to solving the call admission control and dynamic channel assignment problems. This approach is based on the judgment that CAC and DCA can be regarded as an SMDP, and learning is one of the effective ways to find a solution to this problem. A particular learning paradigm has been adopted, known as neuro-dynamic programming (NDP) [reinforcement learning (RL)]. 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. A particular algorithm that appears to be suitable for these two tasks is called Q-learning. In what follows, we briefly describe this algorithm (see [8,9] for more information), and then present the details of how the CAC and DCA problems can be solved by means of Q-learning.

2.1 Q-learning Strategy

The agent, the environment it interacts with, and the task it has to achieve are the components that define the reinforcement-learning framework (cf. Fig. 1). The interaction between the agent and the environment is continuous. On one hand the agent's decision process selects actions according to the perceived situations of the environment, and on the other hand these situations evolve under the influence of the actions. Each time the agent performs an action, it receives a reward. A reward is a scalar value that tells the agent how well it is fulfilling the given task. To be formal let's denote s ($\in S$, a finite set), a representation of the environment's state as it is perceived by the agent, a ($\in A$, a finite set) the selected action, and r ($\in R$, a finite set) the received reward. The agent's decision process is called policy and is a mapping from states to actions (π: $S \rightarrow A$). The interaction between the agent and the environment is continuous and a learning agent modifies its policy according to its experience and to its goal which is to maximize the cumulated rewards over time. The cumulative value $V^{\pi}(s_t)$ achieved by following an arbitrary policy π from an arbitrary initial state s_t is defined as follows:

$$V^{\pi}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_i \qquad (1)$$

where $0 \leq \gamma \leq 1$ is a discount factor. This quantity is often called the *discount cumulative reward* achieved by policy π from initial state s.



s_0
a_0
r_0
s_1
a_1
r_1
s_2
a_2
r_2
…
Goal : learn to choose action that maximize
$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$, where $0 \leq \gamma \leq 1$
environment
action a
reward r
state s
Agent

Fig. 1. The Agent-environment interaction.

A function *Q(s, a)* is used to memorize the expected reward for the action *a* and the state *s*. This function *Q* is called the *action-value* function. The action-value function can be represented either by a look-up table or a function approximator (neural network, regression tree, etc.). On each step of interaction, and in the case of a look-up table representation, the action-value function is updated with equation (2):

$$Q_{t+1}(s,a) = \begin{cases} Q_t(s,a) + \alpha_t \Delta Q_t(s,a), & if \ s = s_t and \ a = a_t \\ Q_t(s,a), & otherwise \end{cases} \tag{2}$$

Where

$$\Delta Q_t(s,a) = \{r_t + \gamma \max_b [Q_t(s'_t, b)]\} - Q_t(s,a) \tag{3}$$

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 [8] 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(s, a)* converges to *Q*(s, a)* with probability 1. The optimal policy $\pi^*(s)$ is the one with the maximum Q-value: $\pi^*(s) = \arg\max_{a \in A(s)} Q^*(s,a)$.

We consider two classes of traffic (C_1 and C_2), but the ideas in this paper can be extended easily to several classes of traffic as well. This cellular system can be considered as a discrete event system. The major events that may occur in a cell include new and handoff calls arrivals and call departures for the two traffic classes. These events are modeled as stochastic variables with appropriate probability distributions. In particular, new call arrivals in a cell obey a Poisson distribution. Call holding time is assumed to be exponentially distributed (cf. Table 2).

2.2 Learning Call Admission Control in FCA system

We consider in this work not only one cell, as we did in a previous one [10,11], but a cellular system with *N* cells. This section develops the dynamic programming formulation suitable for the CAC problem in a cellular network with *N* cells and *L* available channels. The full set of *L* available channels of the system is divided into *S* equal groups each composed of *L/S* channels (cf. Fig. 7). Regular groups of S cells (cluster) are formed such that the frequency reuse distance is maximized. Decreasing *S* (the cluster size) increases the frequency reuse. However, *S* must be large enough to provide sufficient frequency reuse distance and guarantee the required carrier to interference ratio (CIR).

A set of *L/S* channels is permanently assigned to each cell. A new call can be served only if a free channel is available in the set of the cell. If all channels are used, the new call will be blocked and lost even if other channels are available within the frequency reuse area (cluster).

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*, each cell *i* 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=(i,x,e)* defined as:

- $i \in \{1,\ldots,N\}$ is the cell index specifying there is an event *e* occurring in cell *i*.
- $x=(x_1, x_2)$ 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.
- $e=\{1,2,3,4\}$ where

$$e = \begin{cases} 1, & \textit{arrival of a new call of class } C_1 \textit{ in cell } i \\ 2, & \textit{arrival of a new call of class } C_2 \textit{ in cell } i \\ 3, & \textit{arrival of a handoff call of class } C_1 \textit{ in cell } i \\ 4, & \textit{arrival of a handoff call of class } C_2 \textit{ in cell } i \end{cases} \tag{4}$$

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.

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* 1, for each class of traffic. To prioritize handoff calls, larger reward values have been chosen for handoff calls. We also suppose that C1 calls are more important than C2 calls. When a call of class *j* arrives ($e = e_j \in \{1,2,3,4\}$), the reward parameter will be equal to zero when the action is to reject the call (*a=0*) and to η_j when the action is to accept it (*a*=1).

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

η_1	η_2	η_3	η_4
5	1	50	10

Table 1. Immediate Rewards.

In summary, we choose the state descriptor to be $s = (i,(x_1,x_2),e)$, where x_k is the number of calls of class C_k 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 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 agent has to determine a policy for accepting calls given *s*, which maximizes the long-run average revenue, over an infinite horizon. The system constitutes an SMDP with a finite state space $S = \{(i,x, e)\}$ and a finite action space $A=\{0,1\}$ and a finite reward space $R=\{0,1,5,10,50\}$.

2.3 Learning Dynamic Channel Assignment

This work is an extension of a previous work of *Nie and al.* [12]. We consider in this paper not only channel assignment task but also the call admission control problem in a cellular network. We consider a DCA system handling not only one class of traffic as in [12] but two classes of traffic with *N* cells and *M* available channels kept in a common pool. Any channel can be temporarily allocated to any cell, provided that the constraint on the reuse distance is fulfilled (a given signal quality can be maintained). We develop, in this section, the dynamic programming formulation suitable for this problem.

Calls arrive and leave over time and the network can choose to accept or reject connection requests. If the call is accepted, the system assigns to it one of the available channels. The goal of the network operator is the same; He wants to find a DCA policy that maximizes the long-term revenue/utility and reduces handoff blocking probabilities (Contrary to [12] who does not give any priority to handoff calls neither in its simulation nor in the NDP formulation).

The experimental parameters are shown in Table 2. We identify the system states *s*, the actions *a* and the associated rewards *r* as follows:

1) **States:** We define the state *s=(i,A(i),x,e)* as:
- $i \in \{1,\ldots,N\}$ is the cell index specifying there is an event *e* occurring in cell *i*.

- $A(i) \in \{1, 2, \ldots, M\}$ is the number of available channels in cell *i*, which depends on the channel usage conditions in this cell and in its interfering cells[3] *I(i)*.
  To obtain *A(i)*, we define an M-dimensional availability vector u_q for cell *q*, *q*= 1,2,…,*N* as:

  $$u_{qk} = \begin{cases} 0, & \text{if channel } k \text{ available for use in cell } q \\ n, & \text{otherwise} \end{cases}$$

  where n>0.
  By using u_{ik}, *A(i)* can be obtained from

  $$A(i) = \sum_{k=1}^{M} \bar{u}_{ik}$$

  where $\bar{u}_{ik} = \begin{cases} 1, & \text{if } u_{ik} = 0 \\ 0, & \text{otherwise} \end{cases}$
- $x=(x_1, x_2)$ where x_1 and x_2 are the number of calls of each class of traffic (C_1 and C_2 respectively) in cell *i*.
- *e*={1,2,3,4} is the same as in (4).

For the same reasons mentioned above, we do not take into account the states associated with a call departure.

2) **Actions:** We combine the notions of call admission control and channel allocation. Thus, applying an action is to reject the current call request call in cell *i*, or to assign a channel from the *A(i)* available channels to it. So, the possible actions are defined as *A*={0,1,2,…,*M*} where

$$a_i = \begin{cases} k, & \text{accept the call and assign channel } k \text{ to it} \\ 0, & \text{reject the call} \end{cases}$$

where *k* = 1, 2, …, *M* and $u_{ik} = 0$.

3) **Rewards:** The reward *r(s, a)* represents the cost of choosing the action *a* in the state *s*.

$$r(s,a) = \begin{cases} (n_1(a)r_1 + n_2(a)r_2 + n_3(a)r_3) \times \eta_i, & \text{if } a = 1, \ldots, M \text{ and } e = e_i \\ 0, & \text{if } a = 0 \end{cases} \tag{5}$$

When there is a call arrival, the reward parameter will be equal to zero when the action is to reject the call (*a=0*) and it represents the cost of choosing channel *a* to serve the currently concerned call attempt in cell *i* when the call is accepted (*a*≠0). There are many possibilities to define *r*. Here, we consider the type of the call and as in [12] the usage conditions in cochannel[4] cells associated with cell *i*. In the above equation, $n_1(k)$ is the number of compact cells in reference to cell *i* in which channel *k* is being used. Compact cells are the cells with minimum averaging distance between cochannel cells [3]. In the case of a regular hexagonal layout shown in Fig. 7, compact cells are located on the third tier with three cells apart; $n_2(k)$ is the number of cochannel cells which are located on the third tier, but not compact cells in which channel *k* is being used; $n_3(k)$ is the number of other cochannel cells currently using channel k; and r_1, r_2, *and* r_3 are constant associated with the above-mentioned conditions related to $n_1(k)$, $n_2(k)$, and $n_3(k)$ respectively. To obtain $n_1(k)$, $n_2(k)$, and $n_3(k)$ at time *t*, we define the an *M*-dimensional channel status vector for each cell *q*, *q*= 1,2,…,*N* as:

$$s_{qk} = \begin{cases} 1, & \text{if channel } k \text{ is in use in cell } q \\ 0, & \text{otherwise} \end{cases}$$

The parameter η_i represents a constant associated with the type of the current call, and it is defined in Table 1.

In summary, we choose the state descriptor to be $s = (i, A(i), (x_1, x_2), e)$, where *A(i)* is the number of available channels in cell *i*; x_k is the number of calls of class C_k 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 is *A(s)*={0=reject}∪{1,…,*M*} upon a call arrival. Call terminations are not decision points, so no action needs to be taken. The agent has to determine a policy for accepting and choosing the most appropriate channel for calls given *s*, which maximizes the long-run average revenue, over an infinite horizon. The system constitutes an SMDP with a finite state space *S* = {(*i*, *A(i)*, *x*, *e*)} and a finite action space *A*={0,1,…,*M*}.

[3] The set of neighborhood cells that lie at a distance less than a reuse distance *D*.

[4] Cells using the same channel without causing interference.

3 Algorithm Implementation

After the specification of the states, actions and rewards for the two problems, let us describe the online implementations of the Q-learning algorithm for solving them. Here, an important issue arises as to how to store the values of the *Q*-function.

3.1 Q-values representation

A number of powerful convergence proofs have been given showing that Q-learning is guaranteed to converge with probability 1, in cases where the state space is small enough so that look-up table representation can be used. Furthermore, the major difficulty with SMDP problems is the curse of dimensionality (the exponential state space explosion with the problem dimension). Clearly, when the number of state-action pairs becomes large, look-up table representation will be infeasible, and a compact representation where *Q* is represented as a function of a smaller set of parameters using a function approximator in necessary (state aggregation, neural networks [13], regression trees [14]). In a previous work [11] we used a neural network but in this paper we choose state aggregation approximation architecture defined below.

3.1.1 State aggregation

For state aggregation, we consider the partition of the state space *S* into subsets S_0, S_1,…,S_K and introduce a *K*-dimensional parameter vector ϕ whose *m*th component is meant to approximate the Q-value function for all states $s \in S_m$ under action *a*. In other words, we are dealing with piecewise constant approximation

$$\tilde{Q}(s,a,\phi) = \phi(m,a) \qquad\qquad if\, s \in S_m \tag{6}$$

When the value of *K* is small, a lookup table can be used for the aggregated problem. In this case, it can be shown [15] that Q-learning converges to the optimal policy for the aggregated problem. Other function approximators can be used, they may perform well in practice, however, there is no convergence result as for the state aggregation case. In [16,17], the authors show that the performance loss due to state aggregation is bounded by

$$J_0^* - \tilde{J}_0^* \le \frac{2\zeta\varepsilon}{1-\zeta}$$

where J_0^* and $\tilde{J}_0^*$ are the optimal average revenue per unit time for the original and the aggregated problem, respectively. The quantities ζ and ε are defined in [16].

We use feature-based state aggregation to approximate the Q-values, where we learn $Q(S_i,r)$ instead of $Q(s,r)$, where *i* =1,2,…*K*. The number of ongoing calls from each type are aggregated into six levels (*K*=6)..

3.2 Implementation

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.

3.2.1 CAC implementation in FCA system

The flowchart of *Fig.* 2 summarizes the procedures involved in the QCAC-FCA[5] algorithm. We set initial Q-values to random values. When a call arriving (new or handoff call) requires admission to cell *i*, the algorithm first checks if the current free bandwidth of cell *i* can support the call. The call is rejected if the cell does not have enough free bandwidth. Otherwise, QCAC-FCA chooses the action according to

$$a = \arg\max_{a \in A(s)} Q^*(s,a) \tag{7}$$

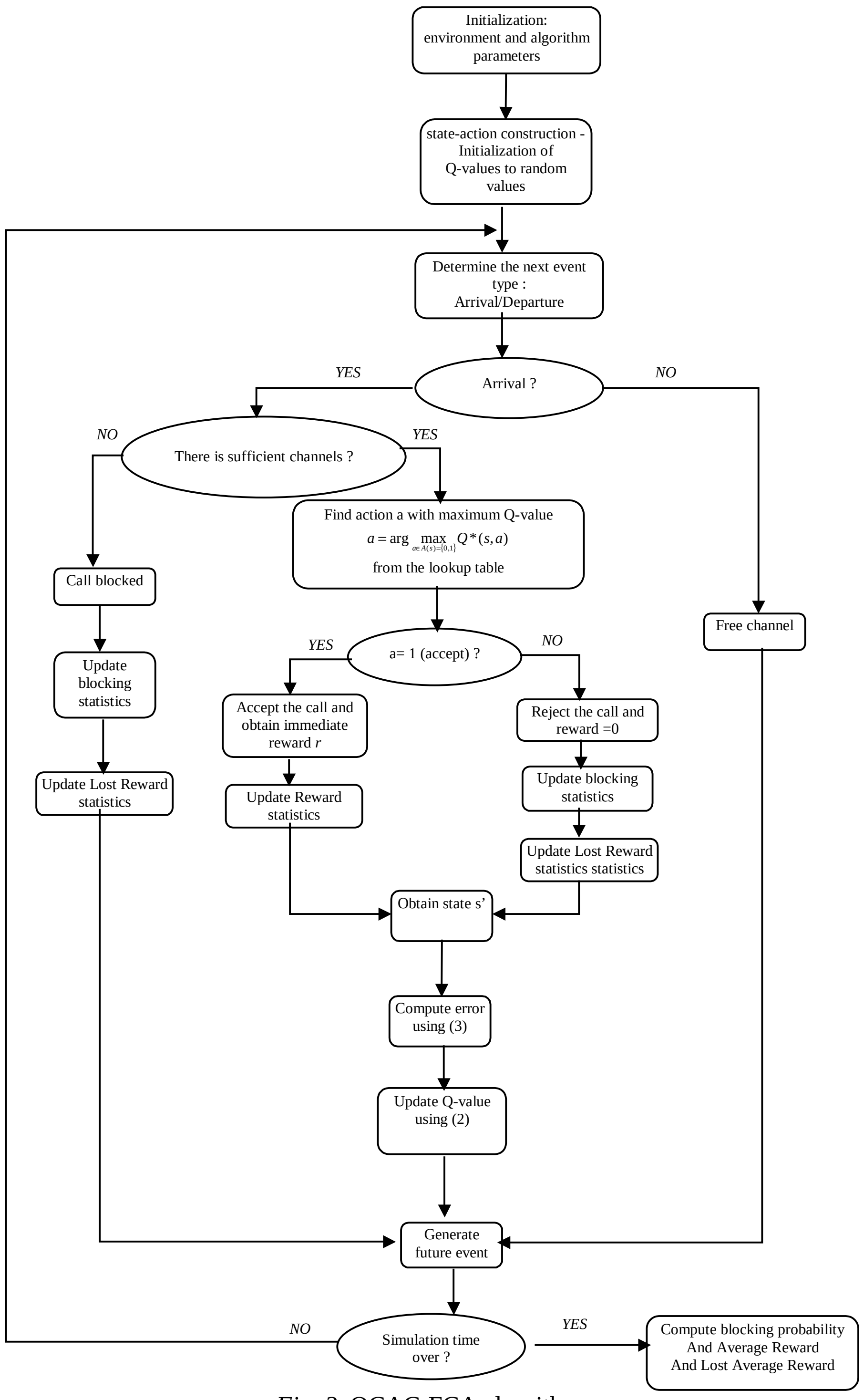
Where *A(s)*={1=accept, 0=reject}.
In particular, (7) implies the following procedures. When a call arrives, the Q-value of accepting the call and the Q-value of rejecting the call are determined from lookup table. If rejection has the higher value, the call is dropped. Otherwise, if acceptance has the higher value, the call is accepted.

To learn the optimal Q-values *Q*(s,a),* the value function is updated at each transition from state *s* to *s'* under action *a* using (2).

We consider a mobile communication system consisting of *N* = 36 hexagonal cells with *L* = 70 available channels in a cluster of *S* = 7 cells because a seven-cell cluster pattern was assumed (cf. Fig. 7). Each cell was assigned L/S = 70/7 = 10 channels. The discount factor is chosen to be γ = 0.5. Training runs typically used a

[5] QCAC-FCA-the online implementation of the Q-learning algorithm for solving the CAC problem in an FCA system

fixed learning rate $\alpha = 0.1$, which seemed to give results even though convergence theorems require decreasing α with time.



Initialization: environment and algorithm parameters
state-action construction - Initialization of Q-values to random values
Determine the next event type : Arrival/Departure
Arrival ?
YES
NO
There is sufficient channels ?
NO
YES
Find action a with maximum Q-value
$a = \arg\max_{a \in A(s)=\{0,1\}} Q^*(s,a)$
from the lookup table
Call blocked
Free channel
Update blocking statistics
YES
a= 1 (accept) ?
NO
Accept the call and obtain immediate reward r
Reject the call and reward =0
Update Lost Reward statistics
Update Reward statistics
Update blocking statistics
Update Lost Reward statistics statistics
Obtain state s'
Compute error using (3)
Update Q-value using (2)
Generate future event
NO
Simulation time over ?
YES
Compute blocking probability And Average Reward And Lost Average Reward

Fig. 2. QCAC-FCA algorithm.

3.2.2 DCA implementation

The flowchart of *Fig.3* summarizes the procedures involved in the Q-DCA[6] algorithm. We set initial Q-values to random values. When there is a call arrival (new or handoff call), the algorithm first determines if accepting this call will violate QoS. If this case, the call is rejected; else the action is chosen according to

$$a = \arg\max_{a \in A(s)} Q^*(s,a) \tag{8}$$

[6] Q-DCA-the online implementation of the Q-learning algorithm for solving the channel assignment and CAC problems in a DCA system.

Where *A(s)*={0=reject, 1,2,…,*M*}.
This formula (8) implies the following procedures. When a call arrives, the Q-value of rejecting the currently concerned call attempt and the Q-values of accepting and choosing channel *a* to serve this call are determined from the lookup table. If rejection has the higher value, the call is dropped. Otherwise, if acceptance has the higher value, the call is accepted and channel *a* is assigned to it.
To learn the optimal Q-values *Q*(s,a)*, the value function is updated at each transition from state *s* to *s'* under action *a* using (2). The parameters in cost evaluation of (5) were r_1 = +5, r_2 = +1, and r_3 = -1. Such a setting would result in a situation in which the channels being used in the compact cells (associated with r_1) have maximum Q-values and thus become the most favorable candidates to be chosen.

We consider a mobile communication system consisting of *N* = 36 hexagonal cells with *M* = 70 channels available in a *pool*. With the reuse distance $D = \sqrt{21}R$, it turns out that if a channel is allocated to cell *i*, it cannot be reused in two tiers of adjacent cells with *i* because of unacceptable cochannel interference levels. Thus, there are at most 18 interfering cells for a specified reference cell. Training runs typically used a discount factor $\gamma = 0.5$ and a fixed learning rate $\alpha = 0.1$.

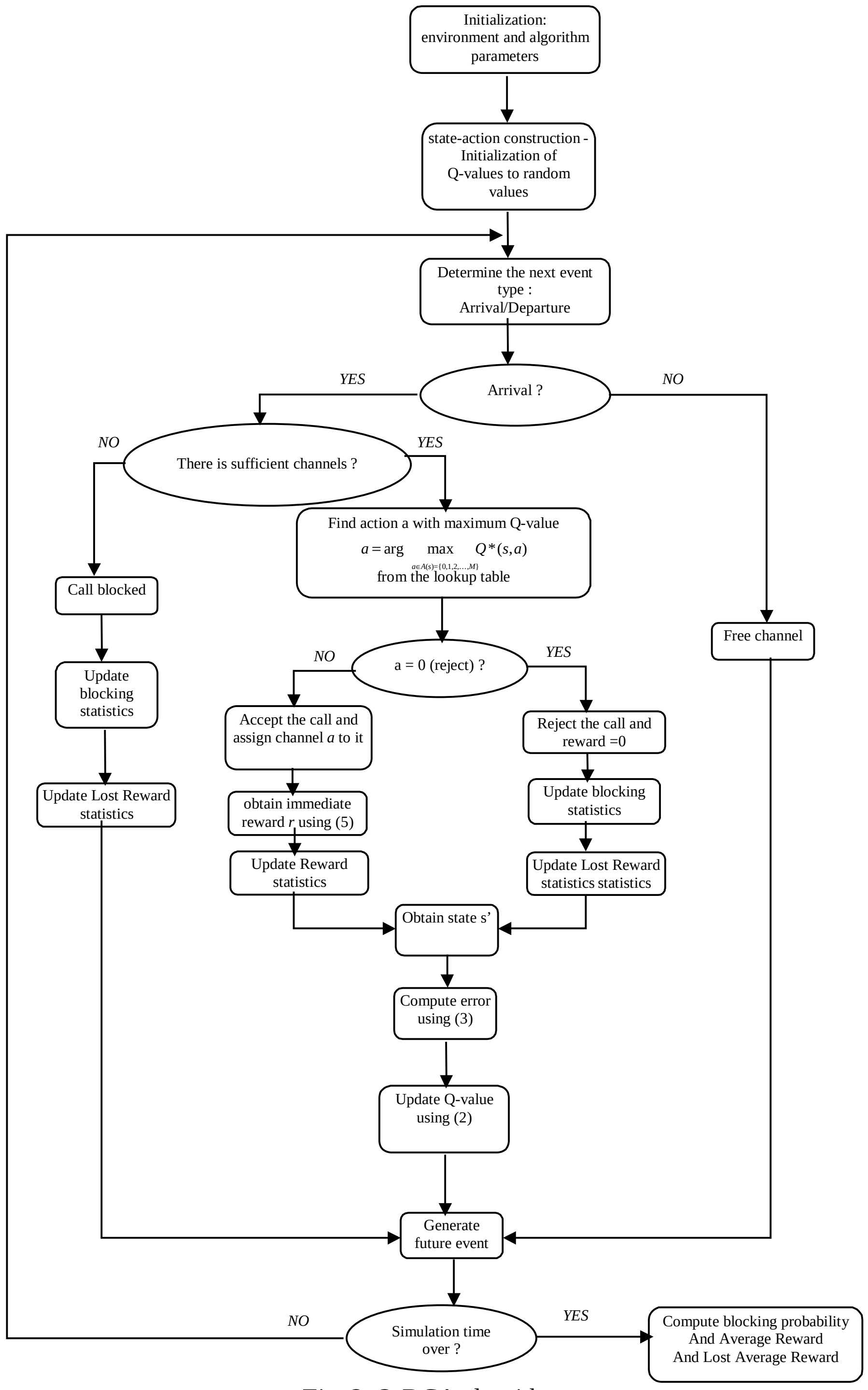

Initialization: environment and algorithm parameters
state-action construction - Initialization of Q-values to random values
Determine the next event type : Arrival/Departure
Arrival ?
YES
NO
There is sufficient channels ?
NO
YES
Find action a with maximum Q-value
$a = \arg \max_{a \in A(s)=\{0,1,2,\ldots,M\}} Q^*(s,a)$
from the lookup table
Call blocked
Free channel
Update blocking statistics
a = 0 (reject) ?
NO
YES
Accept the call and assign channel *a* to it
Reject the call and reward =0
Update Lost Reward statistics
obtain immediate reward *r* using (5)
Update blocking statistics
Update Reward statistics
Update Lost Reward statistics statistics
Obtain state s'
Compute error using (3)
Update Q-value using (2)
Generate future event
Simulation time over ?
NO
YES
Compute blocking probability And Average Reward And Lost Average Reward

Fig. 3. Q-DCA algorithm.

3.3 Exploration

In order for 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 (7), but according to a *Boltzman* distribution [18]. The agent tries out actions probabilistically based on their Q-values using a *Boltzman* or soft max distribution. Given a state s, it tries out action a with probability:

$$p_x(a) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{b\in A} e^{\frac{Q(s,b)}{T}}} \qquad (9)$$

The idea is to start with high exploration and decrease it to nothing as time goes on, so that after a while we are only exploring (*s*, *a*)'s that have worked out at least moderately well before. The *temperature T* controls the amount of exploration (the probability of executing actions other than the one with the highest Q-value). If *T* is high, or if Q-values are all the same, this will pick a random action. If *T* is low and Q-values are different, it will tend to pick the action with the highest Q-value.
At the start, *Q* is assumed to be totally inaccurate[7], so *T* is high (high exploration), and actions all have a roughly equal chance of being executed. *T* decreases as time goes on, and it becomes more and more likely to pick among the actions with the higher Q-values, until finally, as we assume *Q* is converging to *Q** , *T* approaches zero (pure exploitation) and we tend to only pick the action with the highest Q-value:

$$p_x(a) = \begin{cases} 1, & if\ Q(s,a) = \max_{b\in A} Q(s,b) \\ 0, & otherwise \end{cases}$$

4 Experimental results

In order to evaluate the performance of our solutions, we apply a test data set to compare them with the greedy policies (greedy-FCA[8], and greedy-DCA[9]) and the DCA algorithm presented in [12] that we call DCA-Nie. The performance of the algorithms has been evaluated on the basis of the total rewards of the accepted calls (*Total rewards*), the total rewards of the rejected calls (*Total Lost Rewards*), and by measuring the handoff blocking probability.

The parameters used in the training period are given in *Table* 2. The average call duration is assumed to be exponentially distributed with parameter μ_C ($1/\mu_C$ = 180s). The sojourn time of a mobile within a cell is also supposed to be exponentially distributed with parameter μ_H ($1/\mu_H$ = 60s).

	Source Type	
	C_1	C_2
Call duration ($1/\mu_C$)	180 s	180 s
Sojourn time in a cell ($1/\mu_H$)	60s	60 s
Call arrival rate	$\lambda_1 = 120\ calls/hour$	$\lambda_2 = \lambda_1/2 = 60\ calls/hour$

Table 2. Experimental Parameters

A set of simulations was carried out, including the cases of uniform distribution, non-uniform distribution, time-varying traffic load, and equipment failure. The experimental results are shown in *Fig.* 4 through *Fig.* 14. The results show that the reinforcement learning is a good solution for the call admission control problem and channel assignment. The proposed algorithms (QCAC-FCA and Q-DCA) are considerably powerful compared to the greedy policies. 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 are also significantly increased.

4.1 Uniform Distribution

Our first set of experiments involved a constant traffic load for the different classes of traffic among all 36 cells. In this case we used the policy learned in the training period but with eight different traffic load conditions (for both classes C_1 and C_2) among all 36 cells as shown in *Table* 3.

	Sources Type	
	C_1	C_2
Traffic Load (calls/hour)	60	30
	80	40
	100	50
	120	60
	140	70
	160	80
	180	90
	200	100

[7] Initial Q-values were set to random values.
[8] Greedy-FAC: Policy that always accepts a call if the capacity constraint will not be violated by adding this call
[9] Greedy-DAC: Policy that randomly selects a channel to serve a call without any interference measurements. The channels are selected based on a uniform distribution and hence each of the *M* channels has an equal probability of being selected.

Table 3. Experimental Parameters

From the results shown in Fig. 4, we see that the handoff blocking probability decreases significantly using Q-learning compared to the greedy policies for all the traffic loads considered in *Table* 3 and especially when the traffic load is heavy. The handoff blocking probability metric is given by

$$P_{HO} = \frac{\textit{number of handoff calls blocked in the system}}{\textit{number of hndoffs in the system}} \tag{10}$$

Q-DCA and QCAC-FCA outperform the traditional schemes (greedy-FCA and greedy-DCA). Q-DCA performs as well as the DCA-Nie does. We can also notice that Q-DCA algorithm outperforms QCAC-FCA for medium and low traffic loads, but in the presence of congestion there is a trend inversion. Thus, Q-DCA strategy is less efficient than QCAC-FCA under high load conditions in spatial uniform situations.

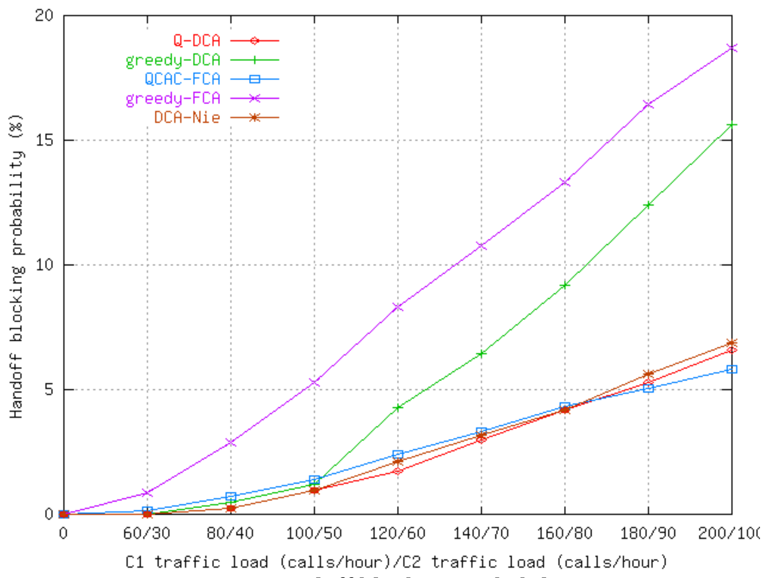
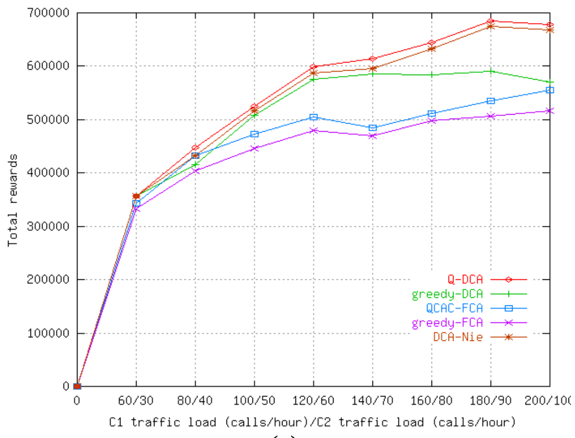
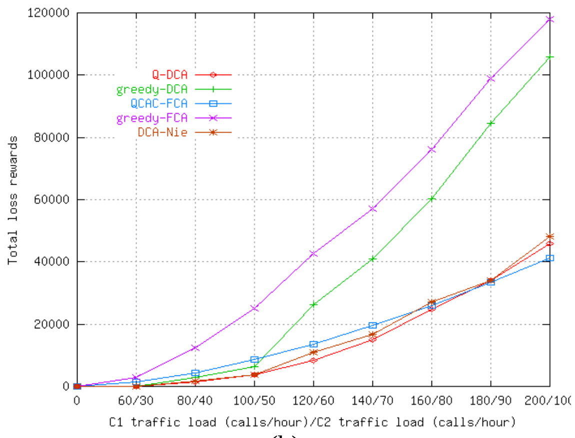


Fig. 4. Handoff blocking probability

Fig. 5(a) shows the total rewards of using learning computed over one simulated hour with the eight different traffic loads of Table 3. 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 cellular network using QCAC-FCA (Q-DCA) are more important compared to those of the greedy-FCA (greedy-DCA) policy. Fig. 5(b) shows that the total loss rewards due to rejection of new calls or the failure of handoff calls were reduced significantly using Q-learning for all the traffic loads and especially when the traffic load is heavy.



(a)



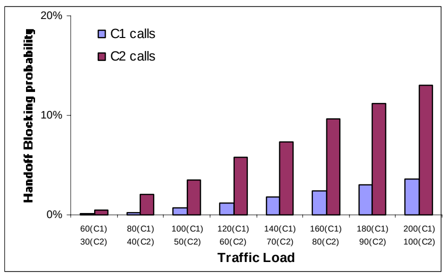
(b)

Fig. 5 (a) Total rewards per 1 hour (b) Total Loss rewards per 1 hour

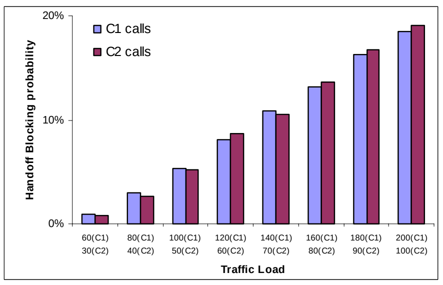
We also compare the handoff blocking probability of C_1 traffic vs. C_2 traffic using QCAC-FCA algorithm (Fig. 6(a)) and Q-DCA (Fig. 6(c)). These metrics are given, for each class of traffic C_i, by

$$P_{HO(C_i)} = \frac{number\ of\ handoff\ calls\ of\ type\ C_i\ blocked\ in\ the\ system}{number\ of\ handoffs\ of\ type\ C_i\ in\ the\ system} \quad (11)$$

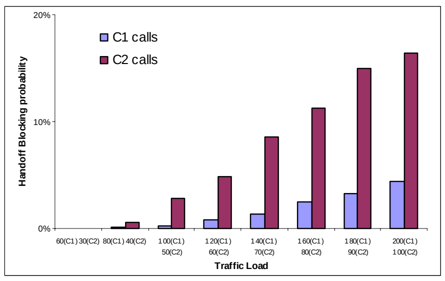
Since C_1 calls have priority than C_2 calls[10], we notice that the handoff blocking probability of C_1($P_{HO(C1)}$) is less than the handoff blocking probability of C_2 ($P_{HO(C2)}$).



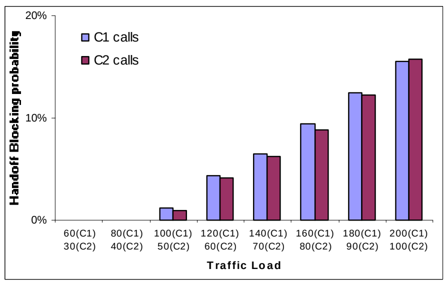
(a)



(b)



(c)



(d)

Fig. 6. Handoff blocking probability of C_1 vs. C_2 using *(a)* QCAC-FCA *(b)* greedy FCA *(c)* Q-DAC *(d)* greedy DCA

This illustrates clearly that QCAC-FCA and Q-DCA have the potential to significantly improve the performance of the system over a broad range of network loads. It is interesting to observe that the Q-values were not relearned and retrained, indicating that the system possesses some generalization capabilities.

4.2 Non-uniform Distribution

In this case we used the policy learned in the training period but the traffic densities in terms of calls/per hour are inhomogeneously distributed among 36 cells (for both classes C_1 and C_2) as shown in Fig. 7. The averaging arrival call rate is 100 calls/per h for C_1 calls and 50 calls/per h for C_2 calls.

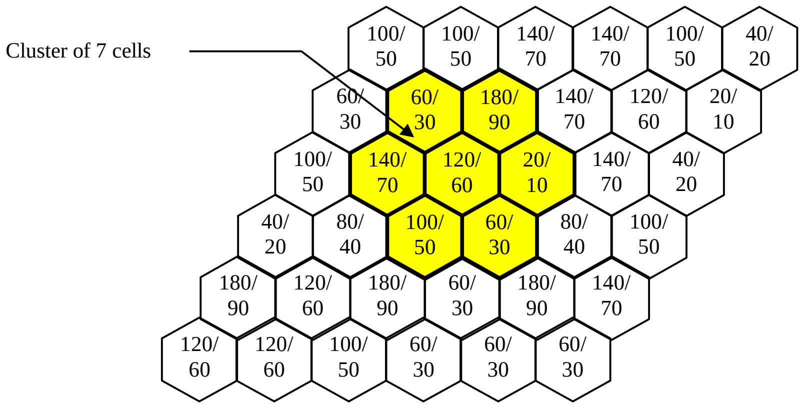


Fig.7. Nonuniform traffic distribution (C_1 calls traffic load/C_2 calls traffic load).

Fig. 8 shows the handoff blocking probabilities of using our methods against the arrival rates which were increased by 100%, 120%, 140%, 160%, 180%, and 200% over the base rates given in Fig. 7. Figures 8 to 10

[10] The rewards associated to C_1 calls are more important than those associated to C_2 calls (cf. Table 1)

indicate significant improvements of the Q-learning algorithms over the greedy schemes. We can also notice that Q-DCA algorithm outperforms QCAC-FCA for medium and low traffic loads, but in the presence of congestion there is a trend inversion. Thus, Q-DCA strategies are less efficient than QCAC-FCA under high load conditions in spatial nonuniform situations.

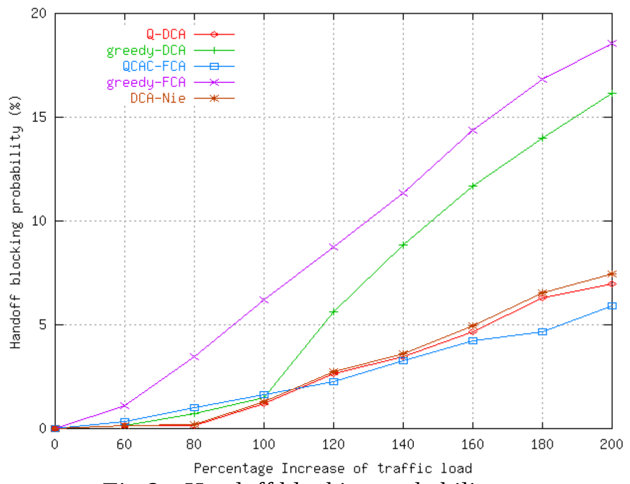
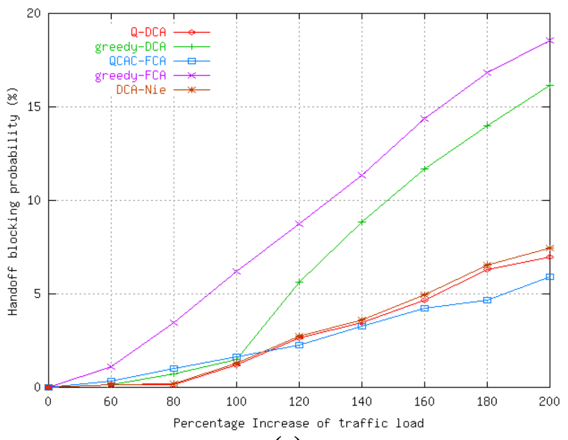
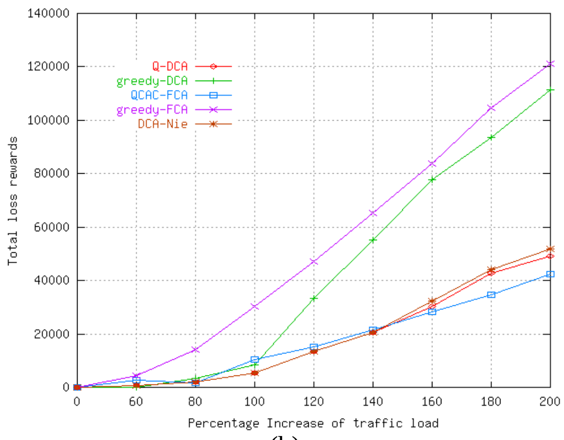


Fig.8 . Handoff blocking probability



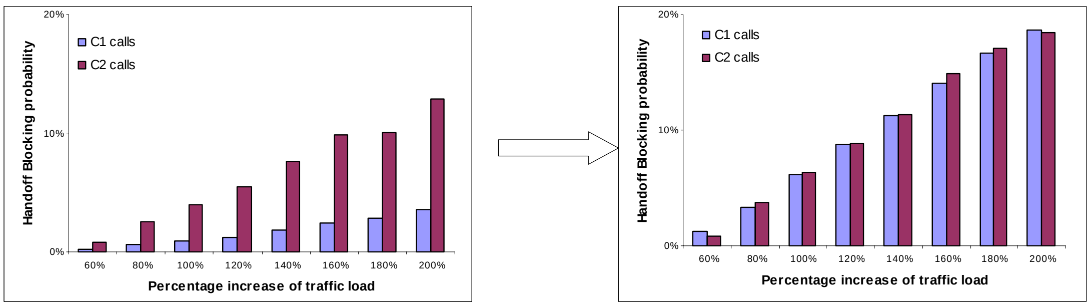
(a)



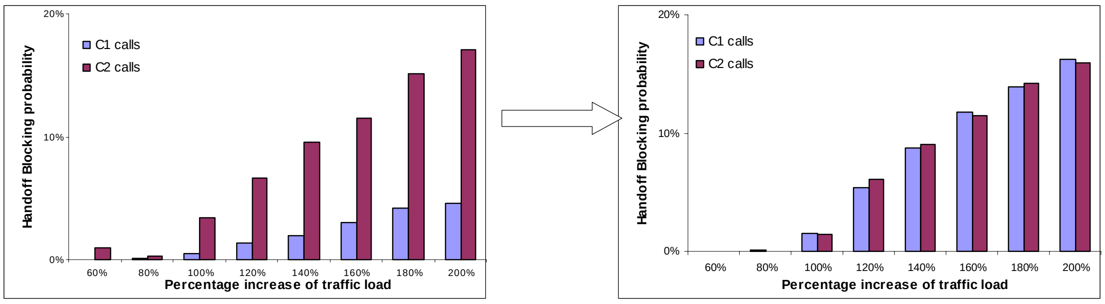
(b)

Fig. 9 (a) Total rewards per 1 hour (b) Total Loss rewards per 1 hour

It is also clear from the Fig. 10 that the handoff blocking probabilities of C_1 calls calculated using (11) are less than the handoff blocking probability of C_2 calls for the two algorithms QCAC-FCA (Fig. 10(a)) and Q-DCA (Fig. 10(c)).



(a) (b)



(c) (d)

Fig. 10. Handoff blocking probability of C_1 vs. C_2 using *(a)* QCAC-FCA *(b)* greedy FCA *(c)* Q-DAC *(d)* greedy DCA

4.3 Time-Varying Traffic Load

The traffic load in a cellular system is typically time varying. In this case, we always use the policies learned in training period and we use, as in [12], the pattern given in *Fig.* 11 concerning arrivals during a typical 24-h business day. The peak hours occur at 11:00 a.m. and 4:00 p.m. *Fig.* 12 gives the simulation results under the assumption that the two traffic classes were spatially uniformly distributed and followed the same time-varying pattern given in *Fig.* 11. The maximum traffic load is set to be 120 calls/h for class C_1 and 60 calls/h for class C_2. The blocking probabilities were calculated on an hour-by-hour basis. The improvements of the proposed reinforcement learning algorithms over the greedy policies are apparent specially when the traffic is heavy (at 11:00 a.m. and 4:00 p.m.). The gains due to RL are about 61% for Q-DCA and 71% for QCAC-FCA.

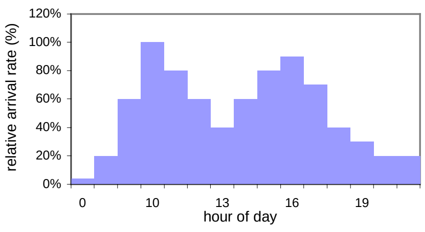


Fig.11. A traffic pattern of a typical business day.

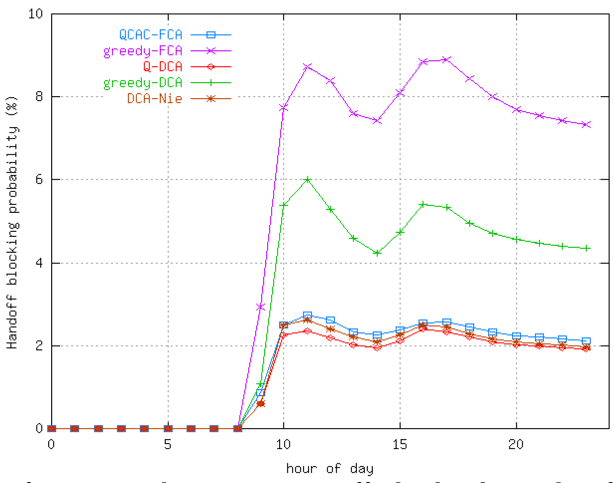


Fig. 12. Performance with time-varying traffic load and spatial uniform traffic.

We also examined the case in which the traffic loads were both spatially nonuniformally distributed and temporally varying. *Fig.* 13 gives the simulation results under the same assumption of the uniform case. The improvements of the proposed reinforcement learning algorithms over the greedy policies are apparent but a more significant improvement was seen in the uniform case. The gains due to RL are about 44% for Q-DCA and 71% for QCAC-FCA.

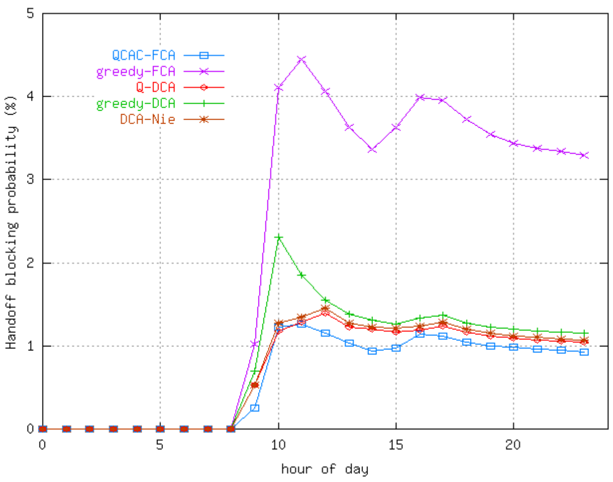


Fig. 13. Performance with time-varying traffic load and spatial nonuniform traffic.

4.4 Equipment Failure in DCA systems

We simulate, as in [12], an equipment failure by some frequency channels being temporally unavailable. There were initially 70 channels available in the system. Between 10:00 and 15:00 o'clock, we temporally shut down 0, 3, 5 or 7 channels. Figure 14 shows the effect of channel failure on the system using Q-DCA algorithm in term of handoff blocking probability. The two classes of traffic were spatially uniformly distributed and followed the same parameters given in table 2. We can clearly see that Q-DCA channel assignment algorithm possesses certain robustness to channel failure situations.

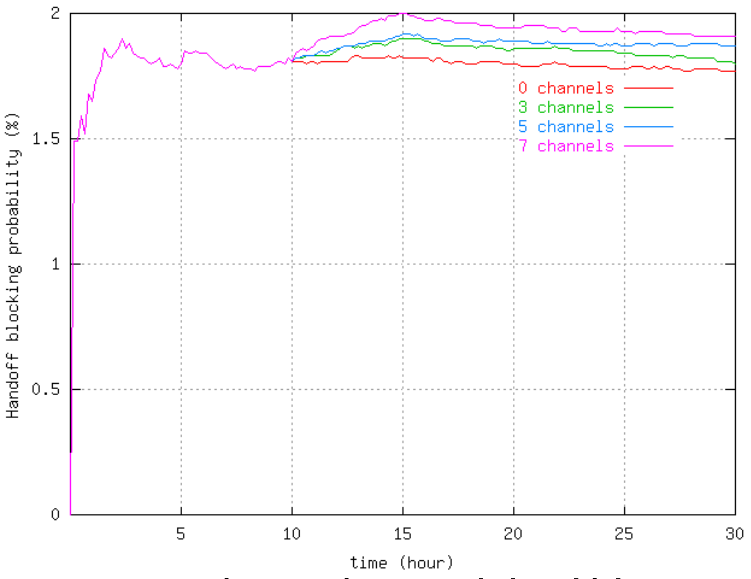


Fig. 14. Performance of Q-DCA with channel failure.

5 Conclusion

In this paper, we presented a new approach to solve two problems in cellular networks. The first one is the channel assignment combined with call admission control problems in DCA system. The second one is the call admission control problem in FCA system. We formulate these two problems 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 solutions are obtained by using a self-learning scheme based on Q-Learning algorithm. The benefits gained by using QCAC-FCA and Q-DCA can be summarized 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. Second, since the proposed schemes are performed in a real-time environment, it is possible to carry out online learning while performing the real admission control and channel assignment. Compared to other schemes, the system offers some generalization capabilities. 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 quality of the system. Third, the channel assignment and acceptation policies can be determined with very little computational effort. Q-DCA algorithm is quite sophisticated compared to other Q-learning channel allocation schemes (DCA-Nie) [12] since it combines the notions of call admission control and channel allocation. It is, also, shown that the proposed algorithms result in significant savings than alternative heuristics.

References

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

[2] M. Zhang and T.S. Yum, "Comparisons of channel assignment strategies in cellular mobile systems," *IEEE Trans. Vehicular Technology*, Vol. 38, pp. 211-215, 1989.

[3] M. Zhang and T.S. Yum, "The nonuniform compact pattern allocation algorithm for cellular mobile systems," *IEEE Trans. Vehicular Technology*, Vol 40, pp. 387-391, 1991.

[4] D. C. Cox and D. O Reudink, "Dynamic channel assignment in two dimensional large mobile radio systems", *Bell Syst. Tech. J.*, Vol. 51, pp. 1611-1627, 1972.

[5] E. Del Re, R. Fantacci, and L. Ronga, "A dynamic channel allocation technique based on Hopfield neural networks", *IEEE Trans. Vehicular Technology*, Vol. 45, pp. 26-32, 1996.

[6] D. D. Dimitrijevic and J. Vucetic, "Design and performance analysis of the algorithms for channel allocation in cellular networks", *IEEE Trans. Vehicular Technology*, Vol. 42, pp. 526-534, 1993.

[7] K. N. Sivarajan, R.J McEliece, and J.W.Ketchum, "Dynamic channel assignment in cellular radio", *Proc. IEEE 40th Vehicular Technology Conf.*, pp. 631-637, 1990.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone, "Classification and Regression Trees", *Chapman & Hall*, 1984.

[15] D.P. Bertsekas and J. N. Tsitsiklis, "Neuro-Dynamic Programming", Athena Scientific, 1996.

[16] 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.

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

[18] C. J. C. H. Watkins, "Learning from delayed rewards," *PhD. thesis*, University of Cambridge, Psychology Department, 1989.

[19] S. P. Singh and D. P. Bertsekas, "Reinforcement learning for dynamic channel allocation in cellular telephone systems," in Advances in NIPS 9, ed. Mozer, M., et al., MIT Press, pp. 974-980, 1997.

[20] 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.

[21] P. Marbach, O. Mihatsch, M. Schulte and J. N. Tsitsiklis, "Reinforcement learning for call admission control and routing in integrated service networks," in Jordan, M., et al., ed. Advances in NIPS 10, MIT Press, 1998.

[22] P. Marbach, J. N. Tsitsiklis, “A Neuro-Dynamic Approach to Admission Control in ATM Networks: The Single Link Case", ICASSP, 1997.

[23] 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.

[24] 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, 1994.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.