# Channel Allocation and Admission Control in Cellular Communications Networks

by

Dimitri A. Papaioannou

Submitted to the

Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements

for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1997

[February 1997]





Author .. .....

Department of Electrical Engineering and Computer Science

June 18, 1997

Certified by... .....

Dimitri P. Bertsekas

Professor of Electrical Engineering

Thesis Supervisor

Accepted by ..... .....

Arthur C. Smith

Chairman, Department Committee on Graduate Students

# Channel Allocation and Admission Control in Cellular Communications Networks

by

Dimitri A. Papaioannou

Submitted to the Department of Electrical Engineering and Computer Science
on June 18, 1997, in partial fulfillment of the
requirements for the degree of
Master of Science

## Abstract

In Frequency Divided Multiple Access cellular communication systems a limited frequency is divided into orthogonal channels that are assigned to mobile subscribers. Channels that are in use in a certain cell site cannot be reused in neighboring cells due to co-channel interference. One of the challenges in providing multimedia wireless services is to maximize the service provided and maintain certain quality measures subject to the restrictions of bandwidth and channel reuse constraints by implementing good call admission and channel assignment strategies. In this thesis, the problem of call admission and channel allocation in a cellular network is formulated as a continuous time controlled Markov process and approximate dynamic programming is used, with conjunction with compact state representation architectures to obtain call admission and channel allocation policies.

Thesis Supervisor: Dimitri P. Bertsekas
Title: Professor of Electrical Engineering

# Acknowledgments

I would like to thank professor Dimitri Bertsekas for his help and guidance in this research and for the inspiration he provided with his enthusiasm for the subject.

I would also like to thank my office mates who have made my stay at LIDS comfortable and enjoyable. I am grateful for their help and their support and most importantly for their friendship. I would also like to thank the members of the NDP group who have guided me with their advice, inspired me with their dedication and team spirit, and shared with me many happy occasions.

I would like to thank my dear friends Marina, Apostolos, Vaggelis, Panagiotis, and Aris who are far in distance but always close to my heart. I would like to thank John, Karin, and Angela for being my family away from home, and Silvia for her constant support.

Finally, I would like to thank my parents Alexandros and Despina and my sister Natalia who have been a source of love and inspiration throughout my life. I owe my every accomplishment to them, including the completion of this thesis.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem Statement

In Frequency Divided Multiple Access cellular communication systems a limited frequency is divided into orthogonal channels that are assigned to mobile subscribers. Channels that are in use in a certain cell site cannot be reused in neighboring cells due to co-channel interference. One of the challenges in providing multimedia wireless services is to maximize the service provided and maintain certain quality measures subject to the restrictions of bandwidth and channel reuse constraints.

The quality of service is typically measured by the probability with which a user is denied service and the probability with which a call is terminated before completion. The first performance measure is called *new call blocking probability* and the second *handoff blocking probability* or *call dropping*. New call blocking occurs when a user attempting a call is denied access to the network, typically because there is no available channel. Handoff blocking occurs when an ongoing call is forced to terminate before completion. This occurs when the mobile subscriber crosses cell boundaries and there is no available channel in the new cell. Handoff blocking is highly undesirable and in many cellular systems a number of channels, called *guard channels* is reserved for handling only handoff calls. A significant challenge is to

design a channel allocation and call admission strategy to minimize handoff blocking while maintaining the new call blocking probability at an admissible level.

In this thesis the call admission and channel assignment problem is formulated as a continuous time Markov decision problem. The number of states is finite but huge, thus the optimal solution is computationally intractable. In order to obtain good suboptimal policies, the cost function of the problem is approximated by a parameterized architecture. The architecture is *trained* by approximately policy iteration via simulation to obtain values for the parameters that will describe good policies.

## 1.2 NDP Methodology

### 1.2.1 The Dynamic Programming Methodology and its Limitations

The theory of dynamic programming provides a unifying mathematical framework for sequential decision-making under uncertainty. Most dynamic decision problems can be cast into a dynamic program formulation. Ideally, one would like to use the theory of dynamic programming to obtain closed-form analytical expressions of an optimal policy. However, because of the nonlinear character of the dynamic programming equations, an analytical solution is often not possible. Although there are iterative numerical algorithms that, in principle, converge to the optimal policy, these algorithms cannot be applied to very complex problems, because the computational burden becomes excessive. This barrier on the successful application of dynamic programming to practical problems was termed by Bellman the "curse of dimensionality", referring mainly to the severe computational requirements when attempting to tackle large-scale, dynamic decision problems. Furthermore, the dynamic programming formulation depends on an accurate mathematical model of the underlying dynamic system and the decision-maker's objectives. However, in many complex problems of practical interest, such a model is not available.

The idea behind approximate dynamic programming is to develop a methodology which allows

the practical application of dynamic programming to problems that suffer either from the curse of dimensionality or by the lack of an accurate mathematical model. This methodology combines the theory of dynamic programming, function approximation techniques, and the use of computer simulation to provide the basis for a rational approach to complex, sequential decision problems under uncertainty.

### 1.2.2 Formulation of the Dynamic Programming Problem, and the Policy Iteration Algorithm

Consider a discrete-time dynamic system evolving under the equation

$$x_{k+1} = f(x_k, u_k, w_k) \tag{1.1}$$

where $x_k$, represents the state, $w_k$ a random disturbance, and $u_k$ a control action chosen from some appropriate set of controls $U(x_k)$.

In the infinite horizon, discounted version of the dynamic programming formulation the objective is to choose the control actions $x_k$ in such a way as to minimize the expected value of an additive discounted cost

$$J^*(x_0) = E\left\{\sum_{k=0}^{\infty} \alpha^k g(x_k, u_k, w_k)|x_0\right\} \tag{1.2}$$

where $\alpha \in (0,1)$ is a discount factor, and $g(x,u,w)$ is the one-stage cost function.

A stationary policy $\mu$ is a function mapping states to controls and the *cost-to-go* of $\mu$ starting at state $x_0$ is defined to be

$$J_\mu(x_0) = E\left\{\sum_{k=0}^{\infty} \alpha^k g(x_k, \mu(x_k), w_k)|x_0\right\} \tag{1.3}$$

It can be shown [1] that if the one-stage costs are bounded then then $J^*(x)$ and $J_\mu(x)$ are the unique solutions of the Bellman equations

$$J^*(x) = \min_{u \in U(x)} E\left\{g(x, u, w,) + \alpha J(f(x, u, w)\right\} \tag{1.4}$$

and

$$J_\mu(x) = E\left\{g(x, \mu(x), w,) + \alpha J(f(x, \mu(x), w)\right\} \tag{1.5}$$

where the expectation is taken with respect to the random parameter $w$.

When the state space and disturbance spaces are finite, it can be shown that the right hand sides of equations (1.4) and (1.5) respectively can be written as

$$(TJ)(i) \equiv \min_{u \in U(i)} \left[g(i, u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j)\right] \tag{1.6}$$

and

$$(T_\mu J_\mu)(i) \equiv g(i, \mu(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\mu(i)) J(j) \tag{1.7}$$

where the states are now represented by integers $i$ and $p_{ij}$ denotes the transition probabilities from state $i$ to state $j$. Also the dynamic programming operators $T$ and $T_\mu$ are defined for convenience.

In this case the following algorithm, known as *policy iteration* can be used to obtain the optimal cost-to-go for the dynamic program.

The algorithm starts with an arbitrary initial policy $\mu_0$ and then performs the following two steps

**Step 1:** Given the policy $\mu^k$, perform a *policy evaluation step* that computes the solution to the linear system of equations $J_{\mu^k}(i) = T_{\mu^k} J_{\mu^k}(i)$, $i = 1, \ldots, n$.

**Step 2:** We then perform a *policy improvement step* which computes a new policy $\mu^{k+1}$ as

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} \left[ \sum_{j=1}^{n} g(i,u) + p_{ij}(u) J_{\mu^k}(j) \right], \quad i = 1, \ldots, n.$$

If $J_{\mu^{k+1}}(i) = J_{\mu^k}(i)$ for all $i$, then the algorithm terminates with policy $\mu^k$. Otherwise, we repeat Steps 1-2 replacing $\mu^k$ with the new policy $\mu^{k+1}$.

### 1.2.3 Compact Representation Methods and Approximate Policy Iteration

Let $J^*(i)$ denote the optimal expected cost from state $i$. The neuro-dynamic programming methodology attempts to approximate this function by a class of functions $\tilde{J}(i,r)$, parameterized by a vector $r$, termed a *compact representation* or an *architecture*. If the class of functions described by the parameterization is wide enough then, by adjusting the parameters $r$ through *training*, a good approximation to the optimal objective function may be obtained. The measure of closeness to the optimal cost-to-go is measured by the sup norm

$$\|\tilde{J} - J^*\| = \sup_i |\tilde{J}(i) - J^*(i)|$$

and is referred as the *power* of the architecture. Unfortunately, in most cases, the power of the architecture cannot be known beforehand. The parameters of the architecture are updated by the following algorithm known as *Approximate Policy Iteration.*

### Approximate Policy Iteration

Approximate Policy Iteration, like normal policy iteration, consists of two stages: *policy evaluation* and *policy improvement.* The policy evaluation step is performed approximately via Monte-Carlo simulation and the policy improvement step is obtained by training the weight vector $r$. The procedure works as follows:

Given a policy $\mu$, the cost-to-go $J_\mu$ associated with this policy is not known. Instead, we have an approximation architecture provided by the function $\tilde{J}(i, r_\mu)$ evaluated at a specific set of parameters or *weights* $r_\mu$. The Approximate Policy Iteration algorithm works as described below:

First, an initial set of parameters $r_0$ is chosen, possibly parameterizing some known initial policy. Then the following steps are performed.

**policy improvement** An initial state $i_0$ is chosen, where by trajectories are generated as follows: From each state $i$, an action is chosen from the set of available actions $U(i)$ by computing

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} g(i,u) + \sum_j p_{ij}(u)\tilde{J}(j, r_{\mu^k}))$$

Then the next state is selected according to the probabilities $p_{ij}(\mu^{k+1}(i))$, and following the state transition from $i$ to $j$, the cost $g(i, \mu^{k+1}(i))$ is incurred. Let the number of steps starting from a state $i$ until trajectory termination be denoted by $M(i)$ ($M(i)$ may be deterministic or a random variable). For each state $i$ that is visited during the trajectory, a cost sample $c(i,m)$ is computed, where $m$ denotes the $m$th simulation run.

**policy evaluation** When a satisfactory number of cost samples is obtained for each state visited, the new policy is evaluated by a policy evaluation step, which amounts to computing appropriate values for the parameter vector $r_{\mu^{k+1}}$

Let $\tilde{S}$ denote the set of states that has been visited, and $M(i)$ the number of cost

samples that have been obtained for each state $i$. A way of obtaining the set of weights corresponding to this new policy is by solving the Least Squares problem

$$\min_{r} \sum_{i \in \tilde{S}} \sum_{m=1}^{M(i)} (\tilde{J}(i,r) - c(i,m))^2 \tag{1.8}$$

**Approximate Policy Iteration with TD($\lambda$)**

The linear least squares problem (1.8) can be solved iteratively by the following method. After simulating the $m$th trajectory $i_0, \ldots i_N$, update the weights by the relation

$$r_{m+1} = r_m - \gamma_m \sum_{k=0}^{N-1} \nabla \tilde{J}(i_k, r) \left( \tilde{J}(i_k, r) - \sum_{m=k}^{N-1} g(i_m, \mu(i_m), i_{m+1}) \right) \tag{1.9}$$

where $\gamma_m$ is a suitably chosen step-size.

By defining $d_k = g(i_k, i_{k+1}) + \tilde{J}(i_{k+1}, r) - \tilde{J}(i_k, r)$, the incremental algorithm (1.9) can be written as

$$r_{m+1} = r_m - \gamma_m \sum_{k=0}^{N-1} \nabla \tilde{J}(i_k, r)(d_k + d_{k+1} + \ldots + d_{N-1}) \tag{1.10}$$

The $d_k$s are called *temporal differences*. There is a variant of this iterative algorithm that is motivated by learning methods and is known as TD($\lambda$) . In TD($\lambda$) the temporal differences obtained at earlier stages are discounted by a factor of $\lambda \in [0, 1]$.

$$r_{m+1} = r_m - \gamma_m \sum_{k=0}^{N-1} \nabla \tilde{J}(i_k, r) \sum_{k=m}^{N-1} d_k \lambda^{k-m} \tag{1.11}$$

The iterative algorithm 1.11 can be used in two different scenaria.

1. The parameters $r_m$ are stored in a place in memory different from the place where $r$ is stored and the updates are performed after a satisfactory number of trajectory runs.

2. The parameters are updated after each trajectory run. In this case, instead of $\tilde{J}(i_k, r)$, $\tilde{J}(i_k, r_m)$ appears on the right side of (1.10). It is also possible to update the weights after each state transition. This method, called *Optimistic Policy Iteration*, is actually not a true policy iteration because the parameters are updated *before* there is enough statistical data to evaluate the present policy. Although it is not clear whether this algorithm will produce improved policies and there is no theory to support such an assumption, it has been successful in many practical cases [21].

### Feature-Based Architectures

It was mentioned earlier that finding an appropriate parameterized class of functions $\tilde{J}(i, r)$ is essential for the performance of these methods. A popular choice of architecture is based on *Neural Networks* that have been theoretically shown to be universal approximators for any continuous function defined in some bounded interval [11]. Furthermore, they have known considerable experimental success. The drawback of using Neural Networks on a problem of large dimension is that they get increasingly harder to train as the number of parameters increases.

Another approach is to simplify the state before it becomes an input to the architecture by forming a reduced state representation. A way to construct a reduced representation of the state is through the use of *features*. Features are important characteristics of the state that describe numerically intuitive facts or heuristic policies for the system under question, and aim to capture the prevailing characteristics of the problem. For example, in a game of poker, important features of the state (a hand of cards in this case) would be the number of cards of equal value in the hand, the highest card, the number of cards of the same suit, etc...

Given $k$ features, $f_1, \ldots, f_k$, a reduced state architecture can be defined by making it a function of the features and the weights $\tilde{J}(f(i), r)$. If the features are good, in the sense

that they capture the essential nonlinearities of the problem, then the functional form of the architecture need not be very complicated. Indeed, for many problems, a linear feature-based architecture $\tilde{J}(f(i), r) = \sum f_k(i) r_k$ can give very good results [19], [2].

## 1.3 Thesis Presentation

This thesis is organized as follows. In chapter 2, the nature of the channel allocation and call admission problem in cellular networks is described and a review of related work in the field is presented.

In chapter 3, the dynamic programming approach for solving continuous time Semi-Markov decision problems is described.

In chapter 4 the channel allocation and call admission problem is formulated in the sequential Markov decision framework. Details of the simulation are discussed and experimental results presented.

In chapter 5 the conclusions and suggestions for future work are discussed.

# Chapter 2

# Problem Description

## 2.1 Description of a Cellular System

This thesis attempts to solve the problem of admission control and channel allocation in a cellular communications network. In this chapter the operation of a cellular network is described and related work in the field is described.

A cellular communications system consists of the following units. A *Mobile Telephone Switching Office* (MTSO), which is the central coordinating element. It contains the cellular processor and cellular switch and is responsible for controlling the call processes and handling billing activities.

The *mobile units*, which are mobile telephones containing a control unit, a transceiver, and an antenna.

The *base stations* or *cell cites* which is a fixed interconnected network providing the interface between the mobiles and the MTSO. Each cell site has a control unit, radio cabinets, antennas, a power plant, and data terminals. The base station communicates with the MTSO and the mobile units via wireless data links.

The geographical area within which a mobile unit can communicate with a particular base

station is called a *cell.* Cells have approximately circular shape and are overlapping to ensure the continuity of communications when users are migrating form one cell to another. However, cells are usually depicted as non-overlapping hexagonal regions to facilitate analysis and simulation.

## 2.2 Operation of a Cellular System

A set of fixed set of frequency ranges called *channels* is allocated to each base station. When a mobile unit wants to communicate with another mobile or the base station it sends a call request to the base station with the strongest signal reception (this is usually the nearest base station). The base station attempts to allocate a channel for the call. If the allocation fails the call is rejected. This is termed *new call blocking* or simply *call blocking.* If the call is accepted, the channel allocated to the mobile unit cannot be used in neighboring base stations because of interference between channels termed *co-channel interference.* The largest radius out of which a channel can be reused is called the *reuse distance.* In this thesis only FDMA (*Freequency Division Multiple Access* systems are considered. In FDMA systems channels are orthogonal and there is no interference between neighboring channels that are in use in the same cell.

For a call in progress there are two alternatives. Either the call terminated in the cell where it originated, in which case the channel is freed, or the mobile unit crosses the border of the current cell and moves to neighboring cell. The procedure of crossing cell regions while a call is in progress is termed a *handoff.* When a handoff is requested, the base station of the new cell tries to allocate a channel from the call. If it fails the call is forced to terminate. This kind of call termination is called *handoff blocking* or *call dropping.* Call dropping is much more undesirable than new call blocking.

## 2.3 Performance Measures

## 2.4 Channel Allocation Schemes

Channel Allocation Schemes are divided into three major categories consisting of *Fixed Channel Allocation*, *Dynamic Channel Allocation*, and *Hybrid Channel Allocation.*

### 2.4.1 Fixed Channel Allocation

In fixed channel allocation schemes a set of channels is permanently assigned to each base station such that the reuse constraints are satisfied. In the simplest case, where the traffic throughout the network is uniform, the same number of channels is assigned to each cell. Finding the pattern that maximizes the number of channels for each cell is equivalent to a graph coloring problem. If the traffic is nonuniform then the channel allocation scheme must be such that cells that handle heavier traffic are assigned more cells. Nonuniform channel allocation patterns are discussed in [22].

An alternative to nonuniform pattern allocation are *channel borrowing schemes*, in which cells that have all their nominal channels occupied can borrow a free channel from a neighbor. There are several channel borrowing schemes some of which result in significant improvement in the blocking rate of the system [7], [18].

Fixed channel allocation schemes have the advantage that are simple to implement but the disadvantage that they cannot capture the effects of temporal changes in the traffic pattern of the system.

### 2.4.2 Dynamic Channel Allocation

To overcome the problem of temporal and spatial variations in the traffic pattern of the system, dynamic channel allocation schemes have been proposed. In these schemes there is

no fixed assignment of channels to cell but rather channels are kept in a central pool and are assigned dynamically to new calls that enter the system provided that this assignment does not violate the reuse constraints. Nearly all dynamic channel allocation schemes employ some reward function that is used to evaluate the relative advantage of using each candidate channel [4], [15], [9], [19]. Some dynamic channel allocation schemes employ channel rearrangement, that involves rearranging the channels in a single cell, or in the whole system, to obtain some favorable allocation pattern.

### 2.4.3 Hybrid Channel Allocation

Hybrid channel allocation schemes combine fixed and dynamic allocation by maintaining *fixed* and *dynamic* sets of channels. Fixed cells are assigned according to some uniform or nonuniform fixed channel scheme and the channels in the dynamic set are assigned to new calls according to some dynamic allocation strategy. An important subproblem of hybrid channel allocation is determining the ratio of fixed to dynamic channels which is in general a quantity depending on the traffic load [14], [17], [15].

### 2.4.4 Handoff Prioritizing Schemes

The methods described so far treat new call arrivals and handed off calls in the same way and do not account for the fact that handoff dropping is much more undesirable than new call blocking. Recently a number of handoff prioritizing schemes have been proposed. They can be divided into three major categories.

- *Guard Channel Schemes* reserve a number of channels exclusively for handoff calls. The remaining channels can be used to handle both new call arrivals and handoffs. It can be shown [13] that increasing the number of guard channels results in an increase in the blocking probability. Therefore, finding the optimal number of guard channels for each cell is a difficult problem that requires knowledge of the traffic pattern and careful

estimation of the channel occupancy time distribution. The disadvantage of guard channels are similar to the disadvantages of the fixed channel assignments schemes in that they are unable to capture spatial and temporal variations in the system traffic.

- *Handoff Queuing Schemes* employ measurements of the power level received by the base station to decide whether a call is about to handoff into a new cell. If the new cell does not have any free channels the call is queued. The call remains queued until a channel in the neighboring cell is freed or the power level drops below a certain threshold in which case the call is dropped. Several handoff queuing strategies have been proposed and some analytical results obtained for special cases [12], [20].
- *New Call Queuing Schemes* were proposed for the reason that new call attempts are less sensitive to delays than handoff call attempts. A number of guard channels is reserved for handling handoff traffic. When a new call arrives and all the non-guard channels are blocked, the new call is queued. Admission strategies that use new call blocking result not only in a decrease in the handoff blocking rate but also in an increase in the total traffic carried by the system [10] [1],[6].

All the strategies discussed employ either a fixed number of guard channels or queuing of handoffs. As mentioned in the description of guard channels, selecting the correct subset of guard channels is critical for the performance of the strategy. However, the optimal number of guard channels can be decided only for a fixed traffic pattern, therefore the performance will decrease if the traffic pattern of the network changes. For handoff queuing schemes reserving guard channels is not necessary, but additional information is necessary, like the measurements of the power of the mobiles signal and maybe of the rate at which the power is decreasing. As a conclusion, there are no good handoff prioritizing strategies for systems with varied traffic where no power measurements are available. This fact indicates that the problem is extremely challenging and suggests that it may be difficult to find good features to form a compact state representation.

[1] This paper contains some errors that are discussed and corrected in [5]

# Chapter 3

# Continuous Time Controlled Markov Processes and Dynamic Programming

## 3.1 Continuous Time Controlled Markov Processes

A Continuous Time Controlled Semi-Markov Process is described by the following.

- A state space $S$, which is assumed to be finite for the purposes of this thesis.
- A set of *actions* or *controls* $U$, together with a collection of subsets of $U$, $U(x)$, that describe the available actions at state $x$. It is useful to introduce the set $K = \{(x,u)|x \in S, u \in U(x)\}$.
- A bounded continuous function $g : K \to R$, that describes the costs incurred.
- A stochastic kernel $Q(.|k)$ on $\mathcal{B}(\mathbb{R}^+) \times \mathcal{S}$ with $k \in K$, called the *transition law.*

  The $k$th transition time is denoted by $t_k$, and we use the notation $x_k = x(t_k)$ and $u_k = u(t_k)$. For any Borel subset $B$ of $\mathbb{R}^+$, $j \in S$, and $(i,u) \in K$ we have

$$Q(B,j|i,u) = P\{t_{k+1} - t_k \in B, x_{k+1} = j | x_k = i, u_k = u\}$$

and $Q$ satisfies the condition

$$\int_0^\infty \tau Q(d\tau, j|k) < \infty$$

for every $j \in S, k \in K$.

Given the initial state $x_0$, the objective is to minimize a measure of the discounted average cost, given by

$$J^*(x_0) = \lim_{N\to\infty} E\left\{\int_0^{t_N} e^{-\beta t} g(x(t), u(t))dt | x_0\right\} \tag{3.1}$$

which can be equivalently written as

$$J^*(x_0) = \sum_{k=0}^{\infty} E\left\{\int_{t_k}^{t_{k+1}} e^{-\beta t} g(x_k, u_k)dt | x_0\right\} \tag{3.2}$$

An *admissible policy* is a sequence of functions $(\mu_0, \mu_1, \dots)$ with $\mu_k : S \to U$, and $\mu_k(x) \in U(x)$. If $\mu_k = \mu \ \forall k$ the policy is called *stationary.* With this notation, the cost of a stationary policy is denoted

$$J_\mu(x_0) = \sum_{k=0}^{\infty} E\left\{\int_{t_k}^{t_{k+1}} e^{-\beta t} g(x_k, \mu(x_k))dt | x_0\right\} \tag{3.3}$$

This problem can be converted into the equivalent discrete time dynamic programming problem [1] in which the costs in (3.2) and (3.3)satisfy the (finitely many) equations

$$J^*(i) = \min_{u\in U(i)} \left[G(i,u) + \sum_j m_{ij}(u) J^*(j)\right] \tag{3.4}$$

and

$$J_\mu(i) = \left[ G(i, \mu(x)) + \sum_j m_{ij}(\mu(i)) J^\mu(j) \right] \tag{3.5}$$

where

$$G(i,u) = g(i,u) \sum_j \int_0^\infty \frac{1 - e^{-\beta\tau}}{\beta} Q(d\tau, j|i,u)$$

and

$$m_{ij}(u) = \int_0^\infty e^{-\beta\tau} Q(d\tau, j|i,u)$$

In some cases there are some instant costs $\hat{g}(i,u)$ incurred when control $u$ is applied. These costs can be included in the above formulation by replacing $G(i,u)$ with $G(i,u) + \hat{g}(i,u)$.

In the special case where the transition interval is exponentially distributed with rate $\nu_i(u)$, the stochastic kernel is separable, the problem is Markovian, and the above equations simplify to

$$G(i,u) = \frac{g(i,u)}{\beta + \nu_i(u)}$$

and

$$m_{ij}(u) = \frac{\nu_i(u)}{\beta + \nu_i(u)} p_{ij}(u)$$

where $p_{ij}(u)$ are the transition probabilities from state $i$ to state $j$ under the action $u$.

## 3.2 Uncontrollable State Components

Now consider equation (3.4) in the Markovian case. Suppose the state $i$ can be written as $i = (a, e)$, where $e$ is a component of the state not affected by the control. In other words, at state $(a, e)$, the next state $(b, \bar{e})$ is determined as follows. First the controllable component is determined by the transition probabilities $p_{ab}(u, e)$, and then the uncontrollable component is determined by the probabilities $q(\bar{e}|b)$. Furthermore, we assume that the transition rates depend only on the controllable state component.

By setting $\hat{J}(i) = \sum_e q(e|a)J^*(a,e)$. Then equation (3.4) can be written as

$$\hat{J}(a) = \sum_e q(e|a) \min_{u \in U(a,e)} \left[ G(a,e,u) + \frac{\nu_a(u)}{\beta + \nu_a(u)} \sum_j p_{ab}(u,e)\hat{J}(b) \right] \tag{3.6}$$

In the problem treaded in this thesis, some more simplifications are possible and will be discussed in the next chapter.

# Chapter 4

# Problem Formulation

## 4.1 Formulation of the Problem as a Markov Decision Process

In this section the call admission and channel allocation problem is formulated as a continuous time controlled Markov process. The discounted cost to be minimized is the negative of the number of users in the system, plus the instant penalties incurred if a call is blocked or dropped.

Assume a cellular network with $N$ cells and $M$ channels. The *state* of the system is fully described by the pair $(A, e)$, where $A$ is the occupancy matrix defined as

$$A_{ij} = \begin{cases} 1 & \text{if channel } j \text{ is used in cell } i, \\ 0 & \text{otherwise.} \end{cases}$$

and $e$ denotes the current event. The current event may be an arrival, a departure, or a handoff request. This component of the state is uncontrollable.

The arrival processes in each cell are independent, Poisson with parameter $\lambda_i$, where $i$ is the cell number. The holding time in each cell, that is the time before either a departure or a

handoff request, is exponentially distributed with mean $1/\mu$ for all the cells. The probability of a handoff from cell $i$ to cell $j$ is given by $p_{ij}$.

The problem belongs to the class of continuous time controlled Markov processes with uncontrollable state components, so equation (3.6) of the previous section applies. The equation can be simplified further because the transition rates do not depend on the control, and the next state is deterministic. In particular, $\nu_A(u) = \nu_A$ and $p_{A\bar{A}}(u) = \delta(\bar{A}, f(A, e, u))$.

where $\delta(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise} \end{cases}$

and $f(A, e, u)$ is a deterministic function that uniquely determines the next state.

With these simplifications equation (3.6) reduces to

$$\frac{\beta + \nu_A}{\beta + \nu} \hat{J}(A) = \sum_e q(e|A) \min_{u \in U(A,e)} \left\{ G(A, e, u) + \frac{\nu_A}{\beta + \nu} \hat{J}(f(A, e, u)) \right\} \tag{4.1}$$

The transition rates $\nu_A$ and the probabilities $q(e|A)$ can be computed by the system parameters as follows:

Define $m_i = \sum_{j=1}^{M} A_{ij}$ = number of channels occupied in cell $i$.

$m = \sum_i m_i$ = number of channles occupied in the system.

Also let $p_i = \sum_k p_{ik}$, where the sum is over the neighbors of cell $i$.

The transition rate out of a state $A$ is given by

$$\nu_A = \sum_{j=1}^{N} \lambda_j + m\mu \tag{4.2}$$

The transition probabilities $q(e|A)$ at each state can be expressed as a function of these numbers.

$$Pr\{\text{Next event is an arrival at cell } k|A\} = \frac{\lambda_k}{\nu_A} \tag{4.3}$$

$$Pr\{\text{Next event is a departure from channel } j \text{ of cell } k|A\} = \frac{(1-p_k)\mu A_{kj}}{\nu_A} \tag{4.4}$$

$$Pr\{\text{Next event is a handoff from channel } j \text{ of cell } i \text{ to cell } k|A\} = \frac{\mu p_{ik}}{\nu_A} \tag{4.5}$$

## 4.2 Motivation and Implementation

Even for relatively small cellular networks, the state space is too large to allow analytical or exact computational solutions. Just a 10-cell, 10-channel system has a state space of approximately $2^{100} \simeq 1.27 \cdot 10^{30}$ states and the systems typically considered are much larger than that. For that reason the problem has been mostly attacked by well thought out heuristics. The idea of obtaining channel assignment strategies by minimizing a cost function has been reported in [9], where a heuristically derived energy function is used and trained by means of a neural network architecture, and in [19] where a Markov decision model is used and the cost-to-go is approximated by a compact representation architecture. This is the approach adopted in this thesis.

The $N$ cells of the system are assumed to be arranged in a two dimensional array of hexagonal blocks, of horizontal dimension $N_h$, and vertical dimension $N_v$. Each cell has an equal number $M$ of channels assigned. Throughout the network there is a fixed reuse distance of $r$ cells. There is no predefined set of channels that a cell can use (as in fixed assignment strategies). Every channel can be used in the cell provided that the reuse constraints are not violated.

Random events are generated from a random distribution with the probabilities described. The control actions available in the present implementation are described below.

- Upon a new call arrival, the available control actions are to accept or reject the call (in the call admission problem) and to allocate a channel if the call is accepted (in the channel allocation problem). The decision to accept is available only if there exists at least one free channel, and the channel assignment decision is subject to the reuse constraints of the network.
- Upon a a call departure, the available set of actions is to perform a channel rearrangement in the cell from which the call departed. The set of rearrangements available is moving a call from an occupied channel to the channel that has just been freed.
- Upon handoff from one cell to the other, the set of decisions is the following. If the call cannot be accepted due to channel unavailability it departs and a channel rearrangement is performed in the cell from which the handoff was initiated. Otherwise, each free channel is considered for accepting the call while simultaneously a rearrangement is performed (if possible) in the cell from which the handoff was initiated.

Given the current event $e$, and a control action $u$ the next state is uniquely determined and denoted $f(A, e, u)$. In the neuro-dynamic programming approach the decision is chosen by the control $u$ that minimizes the approximate cost-to-go.

$$u = arg \min_{u \in U_A} \left\{ g(A, e, u) + \alpha \tilde{J}(f(A, e, u)) \right\}$$

In the above equation $\alpha = \frac{\beta}{\beta+\nu}$.

In the current implementation, the approximation $\tilde{J}$ is a linear architecture of features. Different sets of features were used and compared for both the channel allocation and the admission control problem. A number of heuristic controllers were also implemented for comparison. These controllers use some predefined strategy to select decisions. The performance of the controllers was measured by the blocking and dropping rates attained by the system.

## 4.3 The Channel Allocation Problem without Handoffs

In the simple channel allocation problem without admission control, the handoff probabilities $p_{ij}$ are set to 0 and the only events that can occur are arrivals and departures. The decision set differs from the one just described in that upon a new call arrival the option to reject is no longer available, and the only option considered is the call placement (provided that there are free channels in the cell).

The initial set of features implemented is the one used in [19].

1. An *availability* feature. For each cell $i$ the feature $F_i$ is defined by $F_i(A)$ = number of free channels in cell $i$. This feature attempts to capture the ability of the system to accept new calls.
2. A *packing* feature for each (cell,channel) pair. For each cell $i$ and channel $c$ the feature $f_{i,c}$ is defined by $f_{i,c}(A)$ = number of times channel $c$ is reused in a radius from cell $i$ equal to the reuse distance of the system. This feature attempts to capture the efficiency with which channels are reused in the system.

The results obtained in [19] show that the performance of the approximate dynamic programming method was better than the performance of BDCL which is one of the best available heuristic algorithms [8]. It turns out that the performance of the neuro-dynamic approach is very close to the performance of the heuristic algorithm that selects a channel such that the total number of free channels in the system is maximized. This is one the of best known heuristics for the channel assignment problem and it is used in this thesis for comparison throughout the experiments.

### 4.3.1 Description of Training Parameters and Experimental Results

The first implementation uses the features described and is trained by TD($\lambda$) . The parameters of the implementation are briefly discussed.

**Discount Factor** The discount factor $\alpha$ defined as $\alpha = \frac{\beta}{\beta+\nu}$ is set to 0.995, in order for future rewards to weight considerably. Experiments with lower values of $\alpha$ were also implemented but the results were almost identical.

**Initial State** The initial state is throughout an empty system. It is a non-representative state, but the process transitions very fast from the uninteresting region of a nearly empty system to a nearly full and consequently the cost samples collected in the initial phases are insignificant.

**Initial Weights** Initially, negative weights were used for the features associated with the free channels. It turns out that the initial choice of weights was of small importance in this particular version of the problem.

**Step-size Rule** The step-size rule used is of the form $\gamma_t = \frac{\gamma_0}{t}$ where $\gamma_0$ is an initial value and $t$ is increased by 1 every 1000 iterations. Training is particularly sensitive to the choice of the initial value of the step-size which depends in an unpredictable fashion on the number of features and the size of the system.

**Value of $\lambda$** The default value for $\lambda$ in all the tests is 0. Other values of $\lambda$ where tested and compared.

### Experimental Results for a System with Uniform Traffic

Several simulations were performed for large and small systems under light and heavy traffic conditions. The results presented here are for a 4x4 array of cells with 16 channels in each cell. The reuse distance is 2. The arrival rate is 4 calls per minute uniformly throughout the system and the average call duration time is 2 minutes. The performance measure is the steady state blocking rate of the algorithm. Figure 4-1 illustrates the relative performance of the approximate dynamic programming approach compared to the performance of some heuristics.

The optimal fixed assignment strategy, under which 4 channels are assigned to each cell, does not perform any better than the simplest dynamic assignment strategy which is to assign the

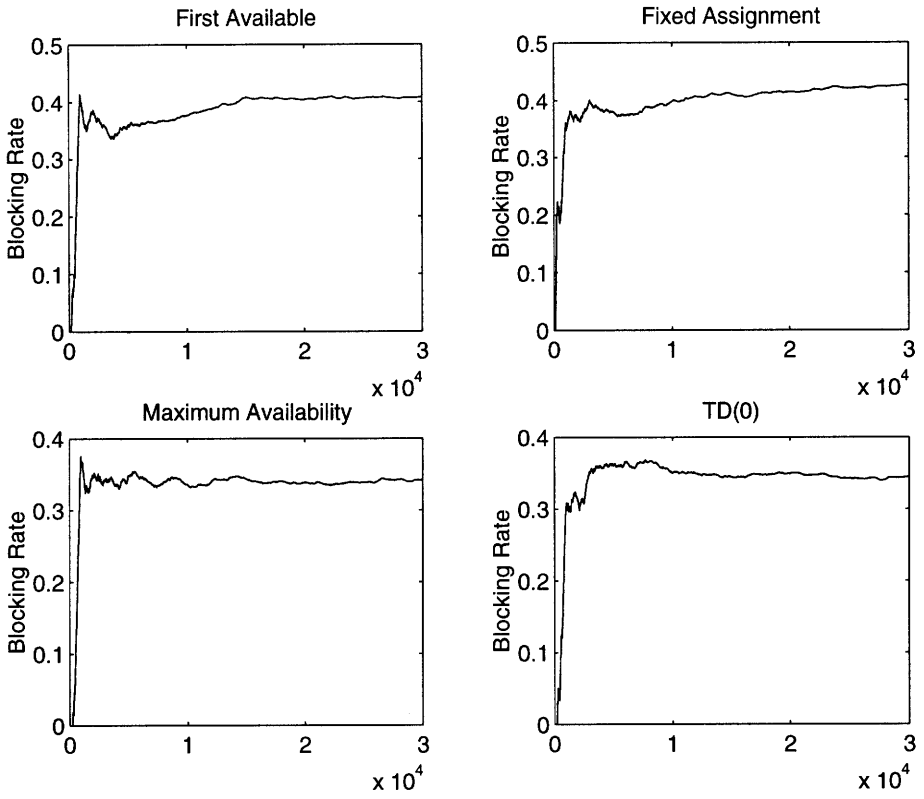


**Figure 4-1**: Comparison of heuristic policies and approximate dynamic programming with optimistic TD(0) for a system with uniform traffic.

first available channel. On the other hand, both the maximum availability heuristic and the linear compact representation architecture trained by optimistic TD(0) perform significantly better. Furthermore, it is seen that their performance is almost identical.

### Experimental Results for a System with Non-Uniform Traffic

Similar experiments were performed for systems with a non-uniform traffic pattern. The system presented here is somewhat larger, consisting of a 5x5 array of cells, with 25 channels each. The reuse distance is 3. The traffic for each cell varies from 1 call per minute to 7 calls per minute. The expected call duration is 1 minute as before.

An important part of the success in the channel allocation problem is played by the clever selection of features. The predominant features in this architecture are the availability features

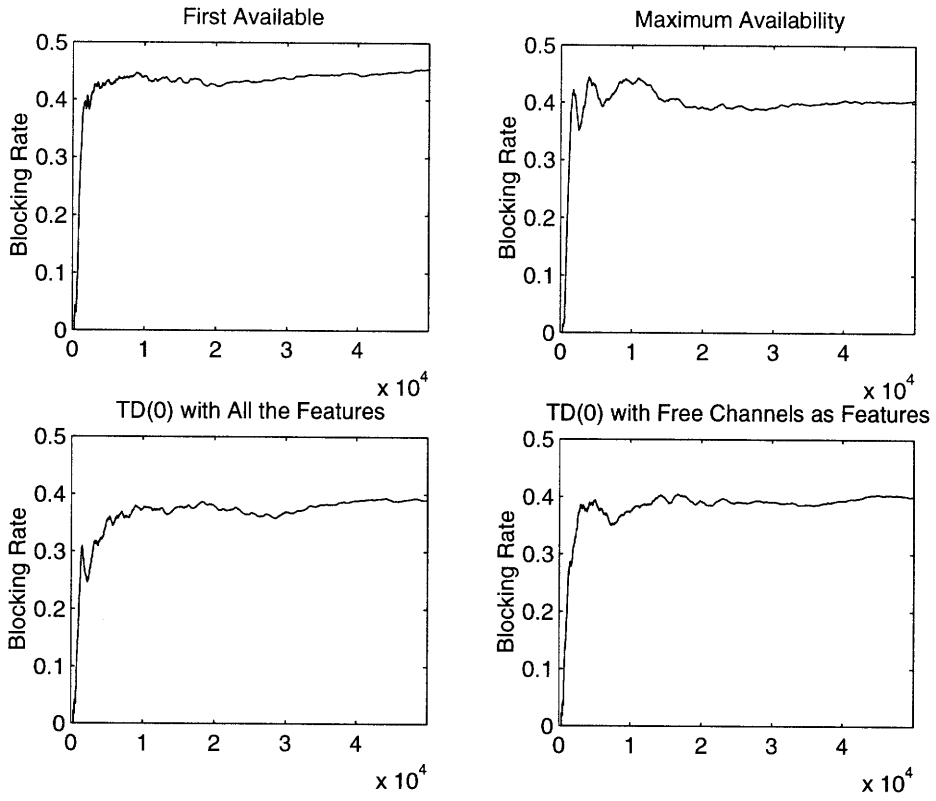


**Figure 4-2**: Comparison of heuristic policies and approximate dynamic programming with optimistic TD(0) for a system with non-uniform traffic.

that count the free channels in each cell. An architecture with only these features should be able to perform as well as the maximum availability heuristic, although the number of parameters is small. This suggests that an architecture with good features can give good performance even if it has only a few parameters. Indeed, this hypothesis was tested by creating an architecture that uses only the free channels per cell as a feature. In figure 4-2 these two architectures are compared with the First Available and the Maximum Availability heuristics.

It is seen that the performance of the maximum availability heuristic and the architecture with the free channels feature are almost identical. The full powered architecture performs only a trifle better.

### Varying the Value of $\lambda$

It is shown in [3] that TD($\lambda$) with a linear architecture converges under the following conditions.

1. The step-sizes $\gamma_t$ are positive, deterministic, and satisfy $\sum_{t=0}^{\infty} \gamma_t = \infty$ and $\sum_{t=0}^{\infty} {\gamma_t}^2 < \infty$.
2. The underlying Markov chain corresponding to the policy being evaluated is aperiodic and has a single recurrent class.
3. There are fewer parameters than states and the features are linearly independent.

This result establishes the validity of approximate policy iteration. The performance bound obtained under the above assumptions deteriorates as the value of $\lambda$ decreases. This suggests that values of $\lambda$ that are close to 1 will attain better performance. There are no convergence guarantees for optimistic policy iteration but it often used in large problems because, when it converges, it converges much faster than non-optimistic policy iteration. The performance varies as a function of $\lambda$ but not consistently with the theory for approximate policy iteration. Determining the best value for $\lambda$ is a matter of trial and error. In the following table, the setup of the non-uniform traffic problem is used to compare different values of $\lambda$.

| $\lambda$ | Blocking Rate |
|---|---|
| 0.0 | .3905 |
| 0.1 | .4111 |
| 0.2 | .4152 |
| 0.3 | .3997 |
| 0.4 | .4016 |
| 0.5 | .3986 |
| 0.6 | .4050 |
| 0.7 | .4150 |
| 0.8 | .4168 |
| 0.9 | .4357 |

It is seen that the best results were obtained by TD(0) and that the performance deteriorates as $\lambda$ gets close to 1.

## 4.4 The Call Admission Problem with Handoffs

In this sections handoffs are introduced explicitly. The handoff probabilities are non-zero and users are allowed to migrate to neighboring cells. It was mentioned earlier that handoff dropping is much more undesirable than new call blocking, therefore it may be advantageous to block some new calls in order to reserve channels for handling handoffs. This introduces the necessity for call admission control.

In this section the admission control problem is studied via simulation. Exact results are obtained for a small system and simulation results are compared for larger systems.

### 4.4.1 Small Systems: Exact Results and Simulation

Exact results can be obtained for a system consisting of two cells. If there are channel reuse constraints the solution is trivial because there will always be a free channel to accept a

handoff (namely the channel from which the handoff departed). Therefore it is interesting to focus on the system without reuse distance constraints.

The study of this small system gives important insights on the form of the optimal policy as well as its dependence on the parameters of the problem. Some insights can also be obtained on things that can go wrong with simulation and training.

The state of the two cell system is described by a pair $(n_1, n_2)$ denoting the number of occupied channels in cells 1 and 2 respectively. Let $g(n_1, n_2)$ denote the reward per state which is the negative of the number of users in the system, i.e. $g(n_1, n_2) = -(n_1 + n_2)$. Let $N$ be the total number of channels in each cell, $c_B$ be the cost of blocking a call, and $c_D$ the cost of dropping. Then for $n_1 < N$ and $n_2 < N$, we have.

$$\begin{aligned} J(n_1,n_2) =& \frac{g(n_1,n_2)}{\beta+\nu_A} + \frac{\lambda_1}{\nu_A}\min\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1+1,n_2), c_B + \frac{\nu_A}{\beta+\nu_A}J(n_1,n_2)\right\} \\ &+ \frac{\lambda_2}{\nu_A}\min\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1,n_2+1), c_B + \frac{\nu_A}{\beta+\nu_A}J(n_1,n_2)\right\} \\ &+ \frac{n_1\mu p_1}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1-1,n_2+1)\right\} + \frac{n_2\mu p_2}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1+1,n_2-1)\right\} \\ &+ \frac{n_1\mu(1-p_1)}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1-1,n_2)\right\} + \frac{n_2\mu(1-p_2)}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1,n_2-1)\right\} \end{aligned} \tag{4.6}$$

If $n_1 = N$ and $n_2 < N$, then

$$
\begin{aligned}
J(n_1,n_2) =& \frac{g(n_1,n_2)}{\beta+\nu_A} + \frac{\lambda_1}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1,n_2)\right\} \\
&+ \frac{\lambda_2}{\nu_A}\min\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1,n_2+1), c_B + \frac{\nu_A}{\beta+\nu_A}J(n_1,n_2)\right\} \\
&+ \frac{n_1\mu p_1}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1-1,n_2+1)\right\} + \frac{n_2\mu p_2}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1,n_2-1)\right\} \\
&+ \frac{n_1\mu(1-p_1)}{\nu_A}\left\{c_D + \frac{\nu_A}{\beta+\nu_A}J(n_1-1,n_2)\right\} + \frac{n_2\mu(1-p_2)}{\nu_A}\left\{\frac{\nu_A}{\beta+\nu_A}J(n_1,n_2-1)\right\}
\end{aligned}
\tag{4.7}
$$

The expression for $n_2 = N, n_1 < n$ is just the symmetric of (4.7), and the expression for $n_1 = n_2 = N$ is easily obtained.

The equations for $J$ are solved exactly by value iteration. The parameters that will be kept constant in the experiment are the following.

- There are 10 channels in each cell.
- The handoff probability from cell 1 to cell 2 is 0.5 and from cell 2 to cell 1 is 0.0.
- The block penalty is 1.0.

The convention in the following experiments is that a value of 2 represents a decision of accepting and a value of 1 a decision of rejecting.

The system described by the following parameters will be used as a point of reference.

| **Parameter** | **Value** |
|---|---|
| Arrival Rate at cell 1 | 2.0 |
| Arrival Rate at cell 2 | 2.0 |
| Block Penalty | 1 |
| Drop Penalty | 5 |
| Alpha | .995 |

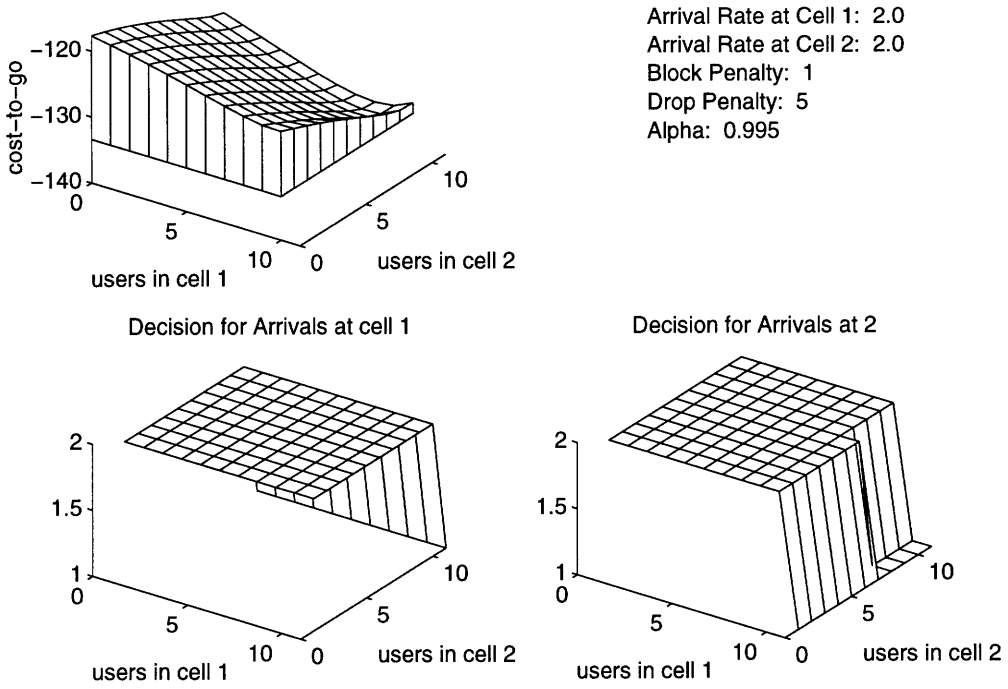


**Figure 4-3**: Illustration of the optimal cost-to-go and optimal policy for a 2-cell system with 10 channels. It is always optimal to accept in cell 1, while the acceptance in cell 2 is a generalized threshold policy.

The optimal policy for this system, illustrated in figure 4-3, is to reserve four channels in cell 2 when cell 1 is fully occupied. All call requests in cell 1 are accepted.

### Varying the Drop Penalty

Recall that the block penalty is fixed at the value of 1. The following table indicates how the reserving policy is modified as the cost of dropping increases.

| Drop Penalty | Channels reserved in cell 2 |
|---|---|
| 1.0 | 0 |
| 2.0 | 0 |
| 3.0 | 0 |
| 4.0 | 0 |
| 4.5 | 3 |
| 5.0 | 4 |
| 6.0 | 5 |
| 7.0 | 6 |
| 8.0 | 7 |
| 9.0 | 8 |

After a certain value the optimal policy changes and the controller starts reserving channels in cell 2 even if cell 1 is not full. Figure 4-4 illustrates the optimal policy for a drop penalty value of 13.

### Varying the Discount Parameter $\alpha$

The effect of varying the discount parameter is what might be expected. As $\alpha$ decreases instant cost becomes more important than cost in the feature therefore the policies become more shortsighted. The following table illustrates the effects of varying $\alpha$.

| Alpha | Channels reserved in cell 2 |
|---|---|
| 0.9 | 4 |
| 0.85 | 3 |
| 0.8 | 1 |
| 0.7 | 0 |

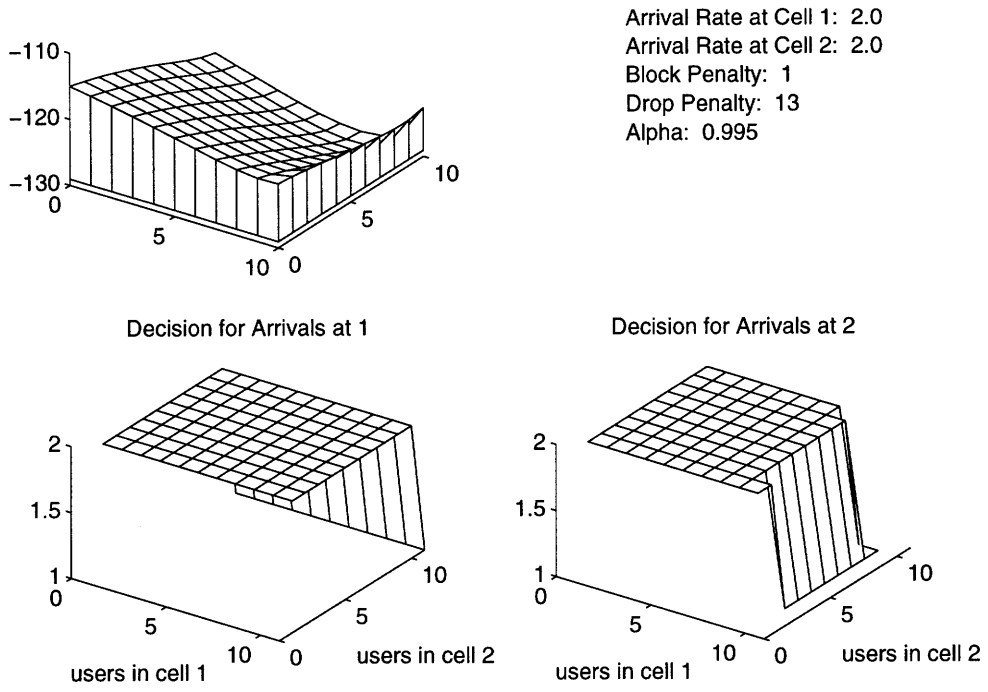


**Figure 4-4**: Illustration of the optimal cost-to-go and optimal policy for a 2-cell system with 10 channels.

### Varying the Arrival Rates

The form of the optimal policy is very sensitive to the traffic pattern of the system. This observation justifies the need for dynamic (adaptable) admission control schemes. In general, it is expected that as the total traffic increases, handoffs become more frequent and more channels should be kept on reserve. Due to the asymmetry in the handoff traffic, the policy is also differs depending on whether the incoming traffic is varied in cell 1 or cell 2. Figure 4-5 illustrates some optimal policies under different traffic patterns. The policy for arrivals at cell 1 is always accept, so only the policy for arrivals at cell 2 is shown.

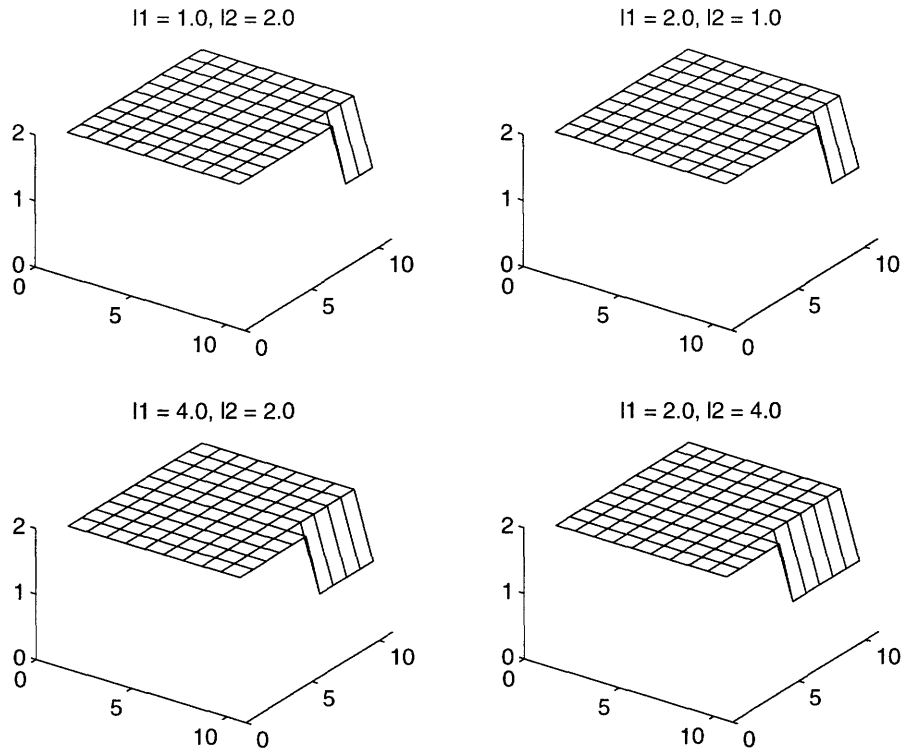


**Figure 4-5**: Illustration of the optimal cost-to-go and optimal policy for a 2-cell system with 10 channels under different traffic patterns. The arrival rate at cell 1 is given by l1 and the rate at cell 2 by l2.

### Some Simulation Results for the Small System

Now some sample simulation runs with this small system are presented. The architecture used is linear in the following features.

- The free channels feature described earlier.
- The times a channel is reused feature also described earlier.
- The probability of handoff in cell 2, which is 0 if cell 1 is empty and 0.5 if it is occupied.

Figure 4-6 shows the considerable advantage of the optimal policy over policies that reserve a fixed number of channels in the second cell irrespectively of the number of occupied channels in cell 1.

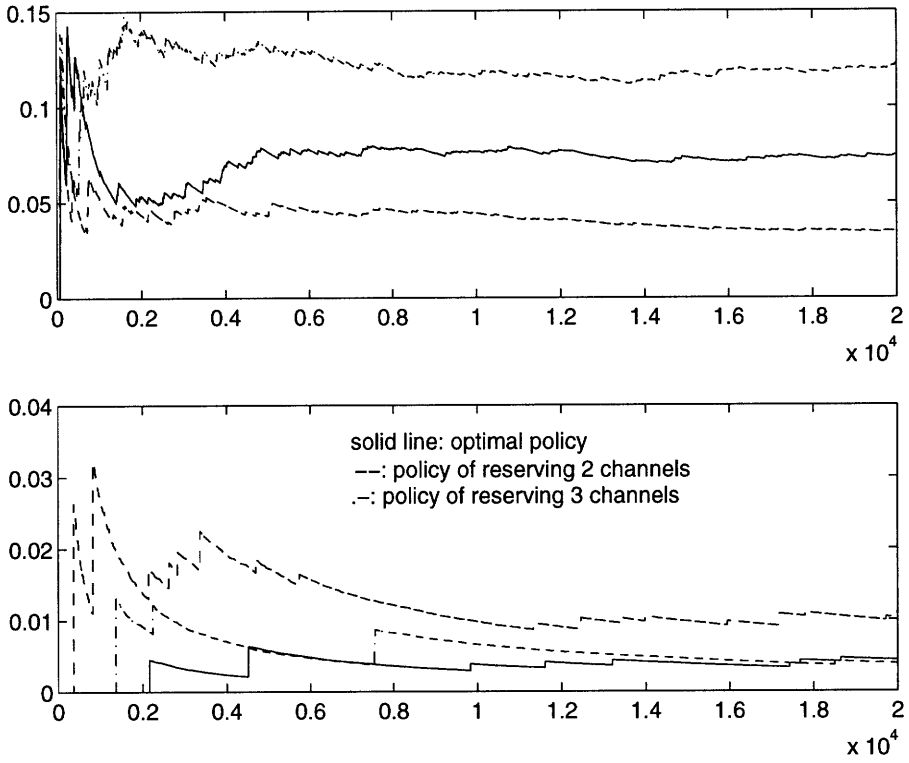


**Figure 4-6**: Comparison of the performance of the optimal policy, against the performance obtained by channel reserving heuristics. The upper graph is the blocking rate and the lower the dropping rate of the system.

A number of experiments are performed in order to evaluate a) the ability of the method to learn a good policy and b) the power of the architecture. In the first experiment, that is designed to test the ability of the method to learn good policies, training starts with the initial weights set at 0. In the second experiment, that is designed to test the power of the architecture, good initial weights are used. In particular, the second experiment works as follows: The controller is using the optimal policy described earlier to make decisions and the parameters (weights) are trained on the decisions made but this policy. Therefore, at the end of the simulation run the final weights must provide a good parameterization of the optimal policy. If the optimal policy is adequately represented then the greedy policy with respect to this parameterization should match the optimal policy. Otherwise, the greedy policy will be worse or just as good.

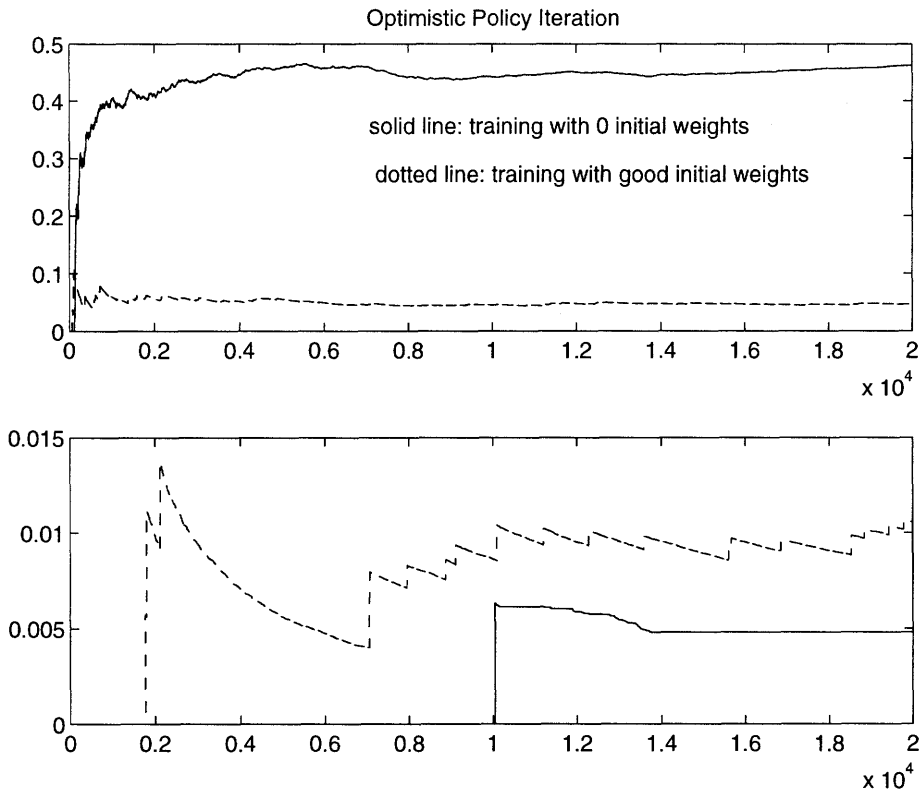


**Figure 4-7**: Results obtained with optimistic policy iteration with different sets of initial weights.The upper graph is the blocking rate and the lower the dropping rate of the system.

Figures 4-7 and 4-8 illustrate the results of these experiments with an architecture trained by optimistic and approximate policy iteration respectively.

The initial policies obtained with both optimistic and approximate policy iterations are way off the optimum. Moreover, the approximate policy iteration oscillates between policies that reject far too often. However, when started with good weights the performance is close to optimal but doesn't match it exactly. This suggests that the ability of the method to learn good policies is weak and that the architecture can adequately, but not exactly, represent the optimal policy.

A few remarks are in order here regarding the performance of non-optimistic policy iteration. The behavior shown in figure 4-8 has been encountered in many simulation runs, in both small and large systems. The reason that approximate policy iteration jumps from to a highly rejecting policy is probably due to the poor initial weights. The reason why it doesn't learn

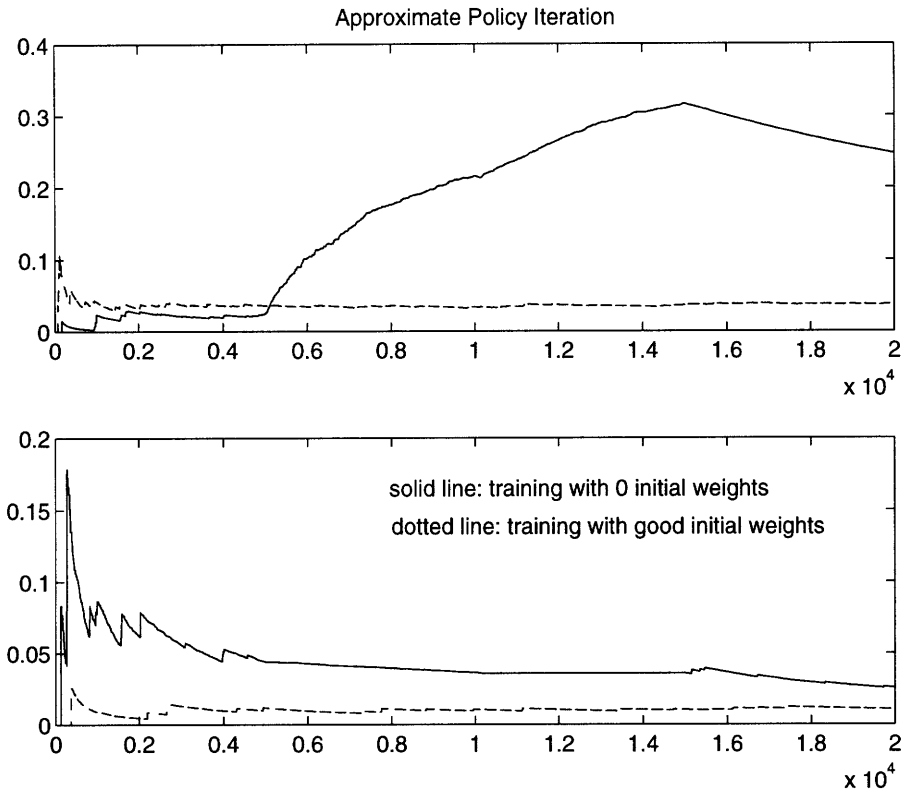


**Figure 4-8**: Results obtained with approximate policy iteration with different sets of initial weights.The upper graph is the blocking rate and the lower the dropping rate of the system.

to perform better is the following: Under a policy that rejects often, states where cells are heavily occupied are never visited. Therefore the samples obtained are only from a limited, non-representative number of states and therefore the cost-to-go approximation obtained is poor and results in a similarly bad policy. Once a highly rejecting policy is reached, the method will go on oscillating between bad policies.

### Conclusions of this Section

The insights gained by the results of this section are summarized here.

1. It appears that under any combination of block/drop penalties, the optimal policy is not to reserve a fixed number of channels in a cell irrespectively of how many channels are occupied in the neighbors. This suggests that better performance than that of simple

channel reserving policies are possible.

2. An increase in the dropping rate results in an optimal policy that reserves more channels.
3. A decrease in the discount parameter results in an optimal policy that reserves less channels.
4. An increase in the total arrival rate results in an optimal policy that reserves more channels, but the exact form of the policy varies non-uniformly as a function of the arrival rates in each cell.
5. Bad initial weights might result in bad policies. Furthermore, approximate policy iteration is likely to get stuck with only bad policies.

### 4.4.2 Large Systems: Comparison of Features and Methodologies

The results of the simulations for the small system are not very encouraging. It appears that both optimistic and approximate policy iterations do not converge to a good policy when started with arbitrary weights. It is possible however that if the simulation is started with a good set of weights an improved policy might be obtained. In addition different sets of features might obtain better performance than the features used in the previous section. However, the 2 cell problem is so small that a lot of important features become irrelevant. Therefore more sophisticated architectures can be tested only on relatively large systems. The cellular system sizes that have been used for testing were 16-cell and 25-cell systems. The results presented in this section are for a 4x4 cell system with 16 channels and a reuse distance of 2.

Handling both the admission control and channel allocation problem with the same architecture turned out to be a very hard problem and all the results obtained by simulations were discouragingly negative. In order to concentrate only on the admission control problem and still have an interesting system (with a 2-dimensional array of cells and non-unity reuse distance) the following decision procedure is used.

Suppose that the system is in state $A$. Upon a call arrival, the channel that will accept the call is decided with the maximum availability heuristic. The approximate cost-to-go of the resulting state, $\bar{A}$, is computed and the call is accepted if $\alpha\tilde{J}(\bar{A}) <$ Blocking Cost $+ \alpha\tilde{J}(A)$. Upon handoffs and departures, the channel allocation and rearrangement decisions are also made with the maximum availability heuristic, so there is no choice of actions when these events occur.

### Features

The following is a list of features that were implemented and their description.

1. **Free channels** This is the feature described in section 4.3, that counts the free channels in each cell.
2. **Number of times a channel is reused** This feature is also described in section 4.3.
3. **Handoff probability** This feature is a measure of the probability of a handoff in a cell $i$. For each neighbor $j$ the number of users $m_j$ is computed and the feature has the value $\frac{\sum_j m_j p_{ji}}{\sum_j m_j}$, where the summation is over the neighbors of $i$.
4. **Drop probability** This feature is the probability that a cell will be dropped in cell $i$, given that the next event is a handoff into $i$. It provides a measure of the danger of an imminent drop.
5. **Used channels** This is just the average reward and can be used either as a cell by cell feature or as a global feature. The reason that the reward has been chosen to serve as a feature is the following: The choice of the magnitude of the drop and block penalties is arbitrary and it might be the case that their accumulative effect is so mush larger than the effect of average rewards, so that the training method does not "recognize" the necessity of keeping a large number of users. In some simulations this feature resulted in a performance improvement.

The architectures compared in this section are composed of combinations of the features described. Other features were also tested but resulted in poor performance.

### Simulation Results for a System with Uniform Handoff Probability

For easy reference the architectures will be denoted by sets of numbers that correspond to the features described. The architectures discussed here are {1,2,3}, {1,2,3,4}, {1,3,4,5}, and {1,4}.

The heuristic against which these architectures are compared is described by the following strategy.

- All allocation and rearrangement decisions are made by the maximum availability heuristic.
- A new call request is rejected if the cell in which the call arrives has less free channels than a prespecified fixed number.

Note that this is a better heuristic than a simple guard channel scheme of the kind described in chapter 2. The reason is that the channels that are reserved are not prespecified therefore better utilization of the available resources is possible. Figure 4-9 shows the performance of this heuristic when the number of reserved channels are 1 and 2. The corresponding heuristics will henceforth be referred to as heuristic 1 and heuristic 2.

Figures 4-10 and 4-11 illustrate the performance of TD(0) with linear architectures with zero initial weighs.

It is seen that different architectures exhibit quite different performance. The performance of some architectures is close to the heuristic 1, while others are close to heuristic 2.

A note on performance is in order here. The absolute values of the blocking and dropping rates are not important as performance measures in this setting. The blocking and dropping penalties are chosen arbitrarily to reflect the fact that dropping is much more expensive than

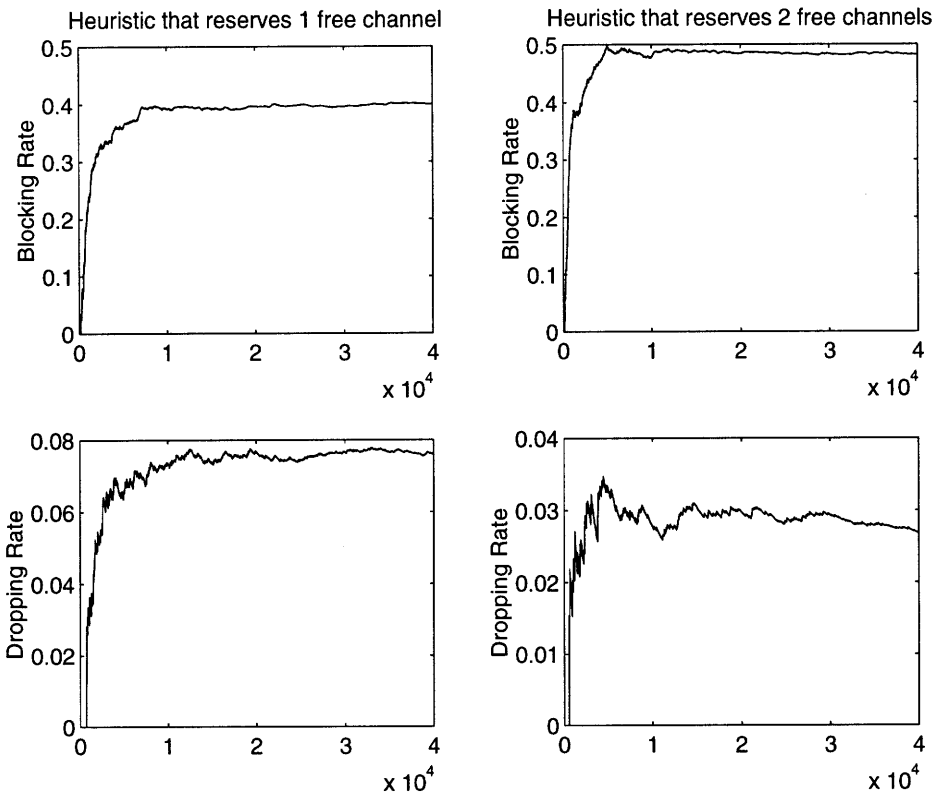


**Figure 4-9**: Performance of heuristic policies.

blocking and they do not correspond to some particular expectation about the blocking and dropping rates of the system. In fact these rates cannot be computed, even for fixed policies, in all but the simplest cases. Furthermore, as discussed in chapter 2, there are no standard performance measures to evaluate the relative blocking/dropping rates obtained. Therefore an absolute rating of the policies compared is impossible. A policy is considered "better" than some other if it attains a lower dropping rate for (about) the same blocking rate, or vice-versa. In particular, one cannot say that heuristic 1 is better than heuristic 2. These heuristics are just reference points against which the solutions obtained by neuro-dynamic programming methods are compared.

With this perspective, it is seen that architectures {1,2,3} and {1,4} are "closer" to heuristic 2, while {1,2,3,4} is "closer" to heuristic 1. So it makes sense to start off training architecture {1,4} with the weights obtained by training on the second heuristic in the hope to obtain an improved policy.

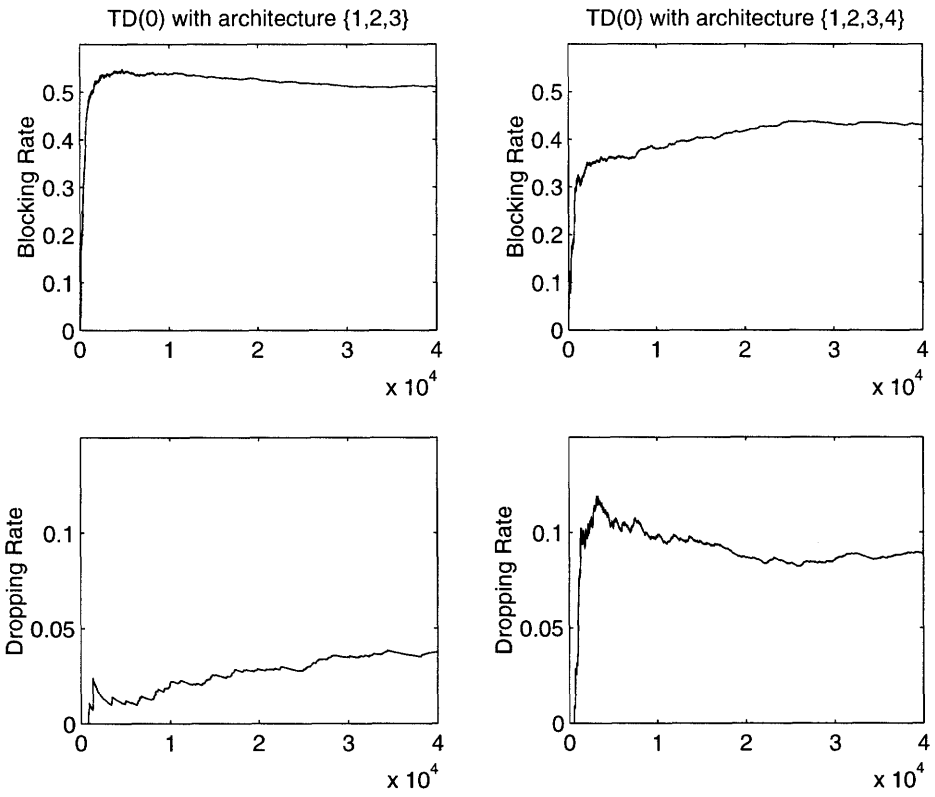


**Figure 4-10**: Performance of certain architectures trained by TD(0) in a system with uniform handoff probability.

The experiment was performed for architectures {1,4} with weights obtained from heuristic 2 and {1,2,3,4} with weights obtained from heuristic 1. Both optimistic and approximate [1] policy iterations were implemented. In the non-optimistic version the weights are updated every 10,000 iterations. The results are illustrated in figures 4-12 and 4-13.

It is seen that better performance is not attained even though good initial weighs are used. In fact, for architecture {1,2,3,4} the approximate policy iteration oscillates.

### Simulation Results for a System with non-Uniform Handoff Probability

The experiments of the previous sections are repeated for a system were the handoff probabilities are non-uniform. Figure 4-14 illustrates the performance of the two heuristics.

[1] The words approximate and non-optimistic are used interchangeably

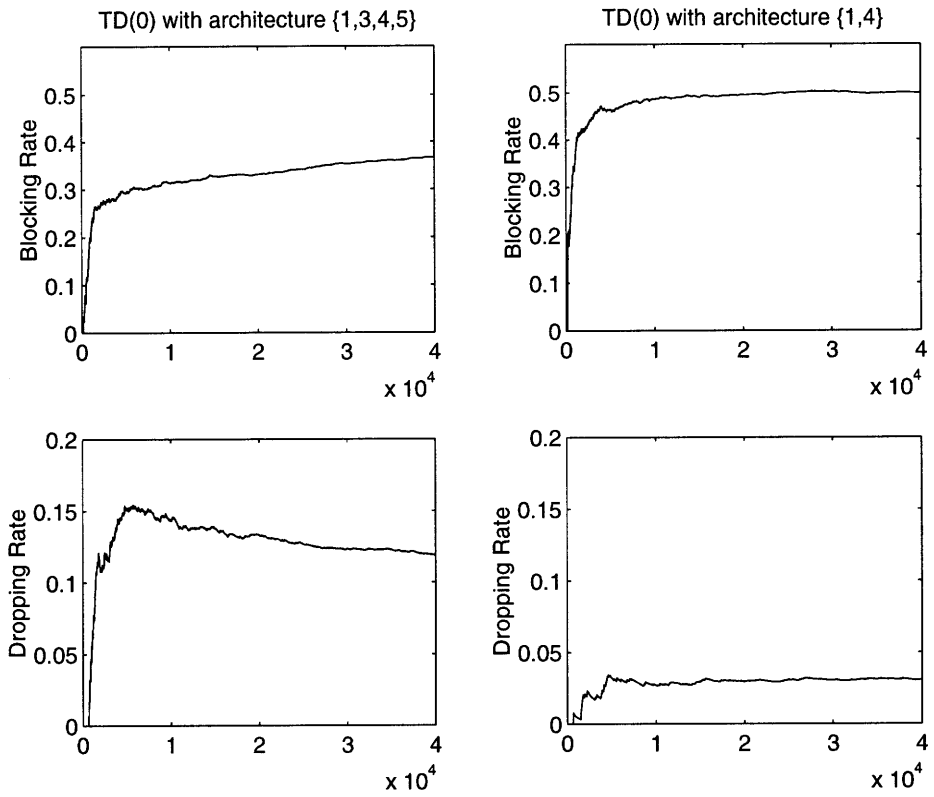


**Figure 4-11**: Performance of certain architectures trained by TD(0) in a system with uniform handoff probability.

Figures 4-15 and 4-16 illustrate the performance of the architectures with 0 initial weights.

Finally, figures 4-17 and 4-18 illustrate architectures {1,4} and {1,2,3,4} respectively trained with good initial weights.

It is seen that the good initial weights do not help, as the resulting architectures perform even worse than the ones obtained before.

## 4.5 Conclusions of this Section

The call admission and channel allocation problem was formulated as a continuous time Markov decision problem and it's solution was attempted by approximate dynamic programming. First, the channel allocation was treated and the results of [19] were duplicated.

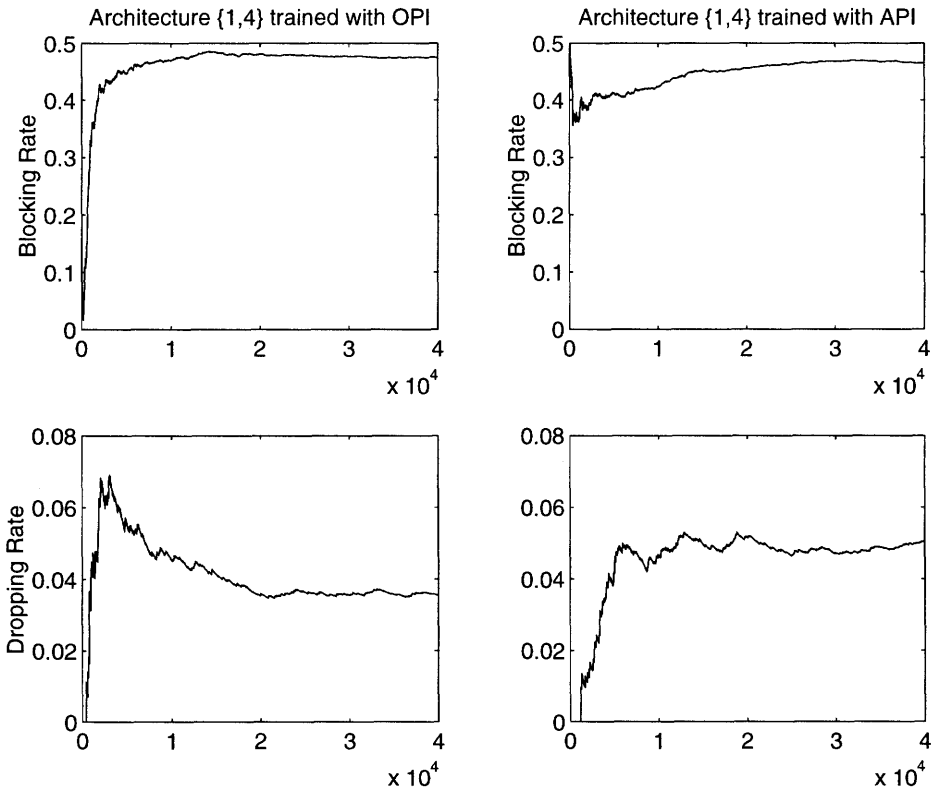


**Figure 4-12**: Performance of TD(0) in a system with uniform handoff probability and good initial weighs for a system with uniform handoff probability.

The admission control problem is exactly solvable for a system consisting of 2 cells. The solution of the two cell problem was used to develop insight for the solution of larger systems and demonstrate the dependence of the optimal solution on system parameters.

A solution of the admission control problem for large cellular networks was attempted by a compact representation method consisting of a linear architecture of features. Several different architectures were used and compared. The policies obtained did not outperform the heuristics with which they were compared although some exhibited comparable performance.

An discussion of the inferences drawn by this results is found in chapter 5.

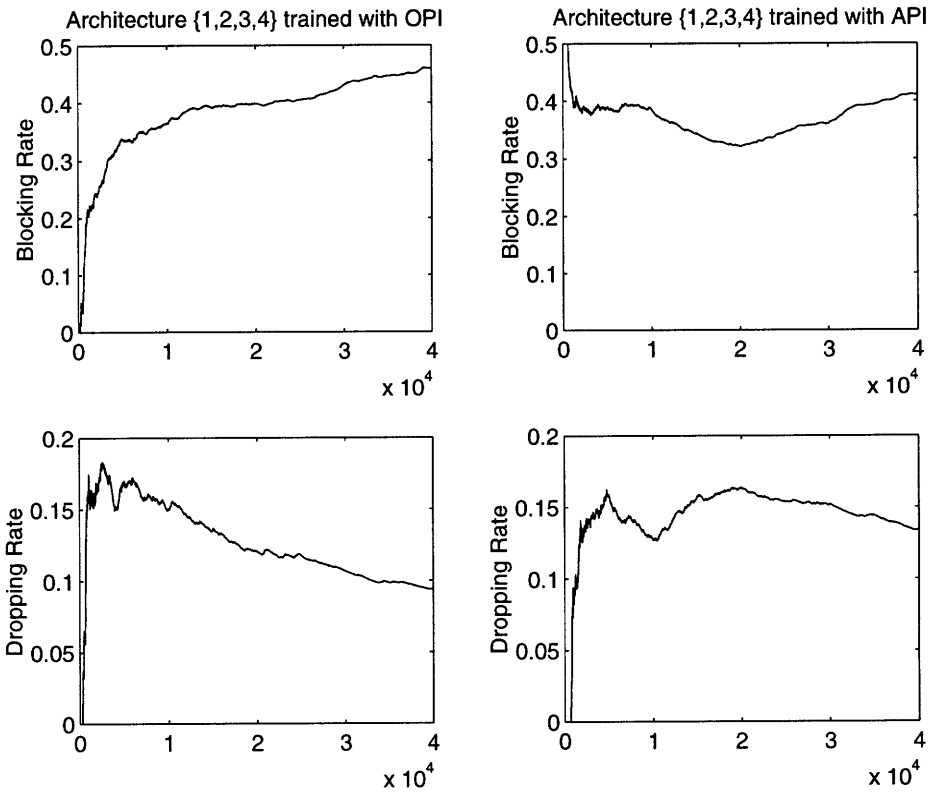


**Figure 4-13**: Performance of TD(0) in a system with uniform handoff probability and good initial weighs for a system with uniform handoff probability.

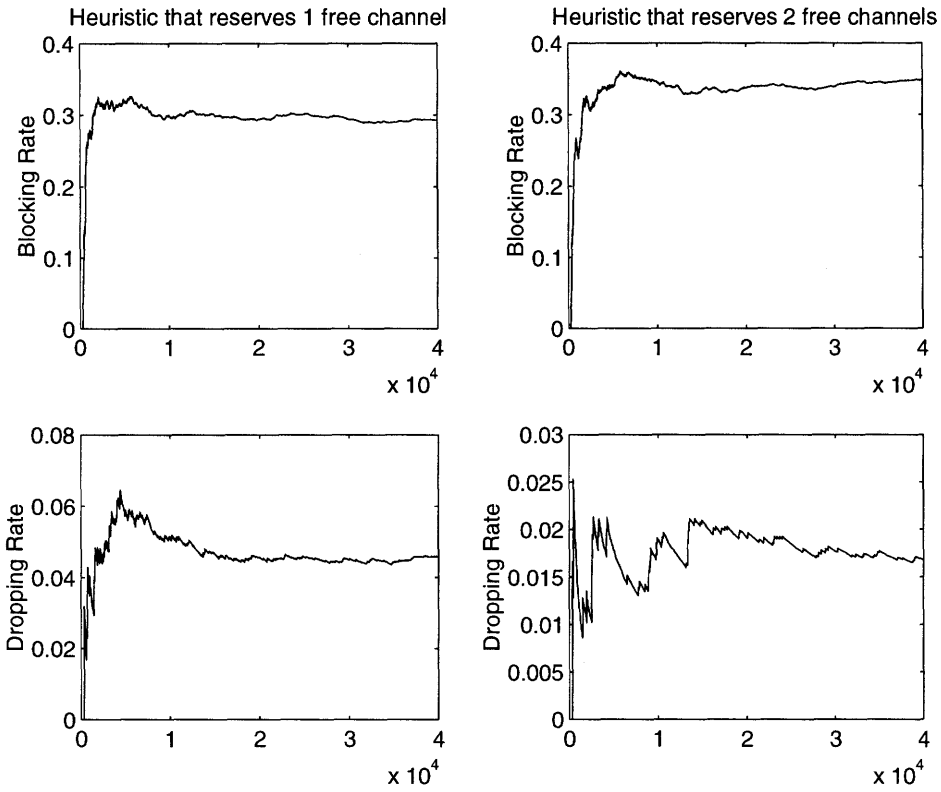


**Figure 4-14**: Performance of heuristic policies in a system with non-uniform handoff probabilities.

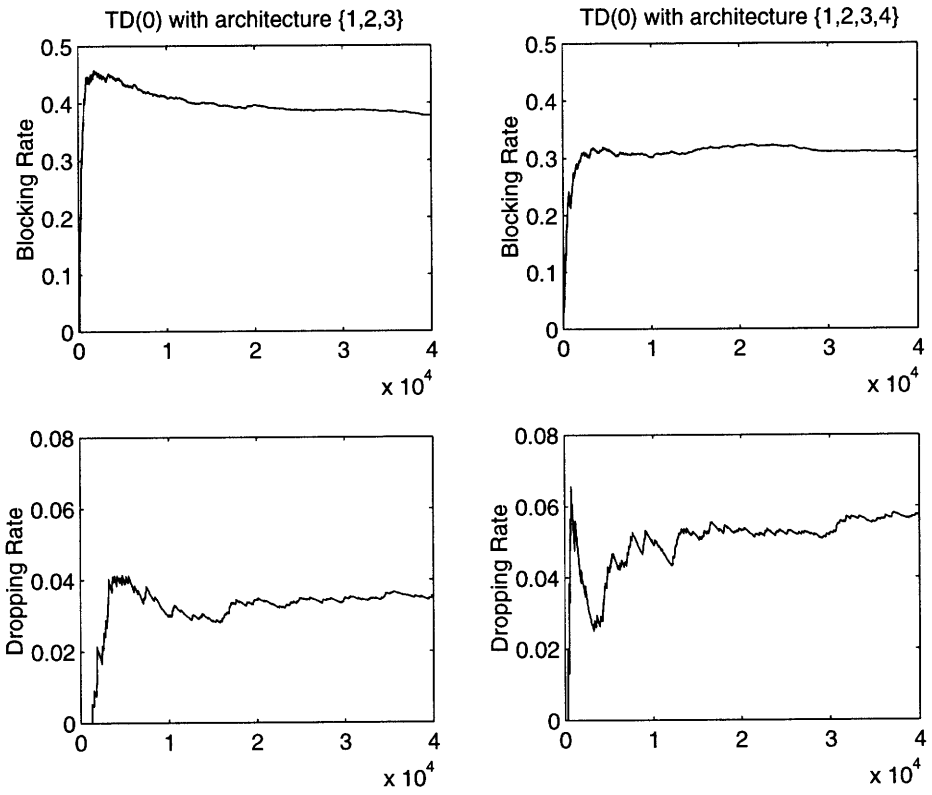


**Figure 4-15**: Performance of TD(0) in a system with non-uniform handoff probability.

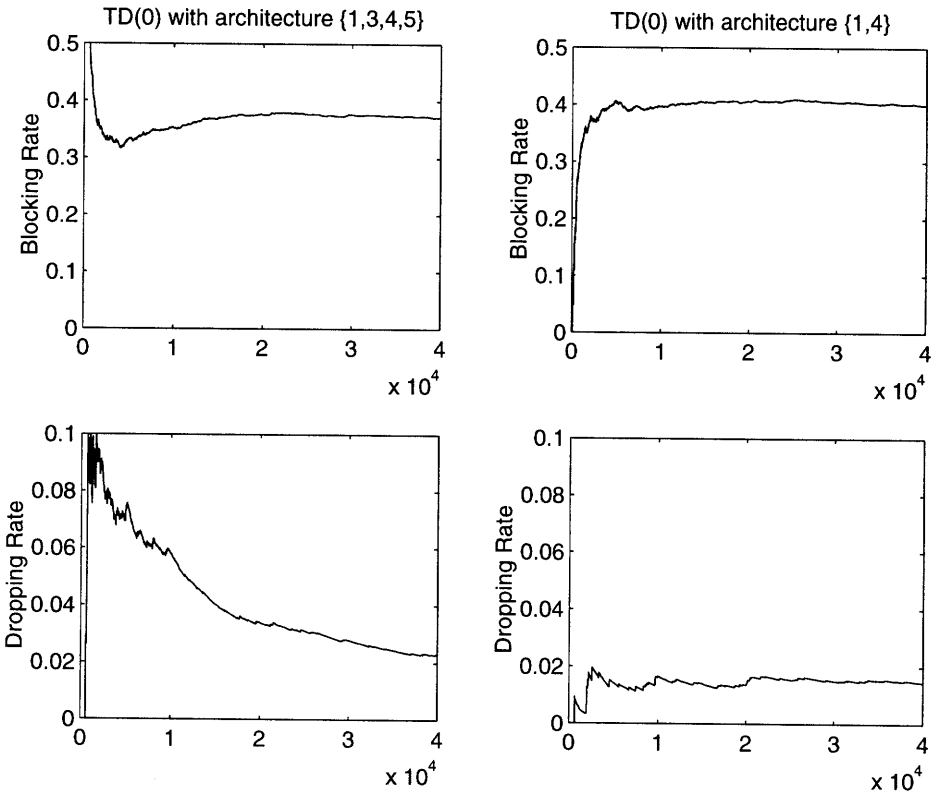


**Figure 4-16**: Performance of TD(0) in a system with non-uniform handoff probability.

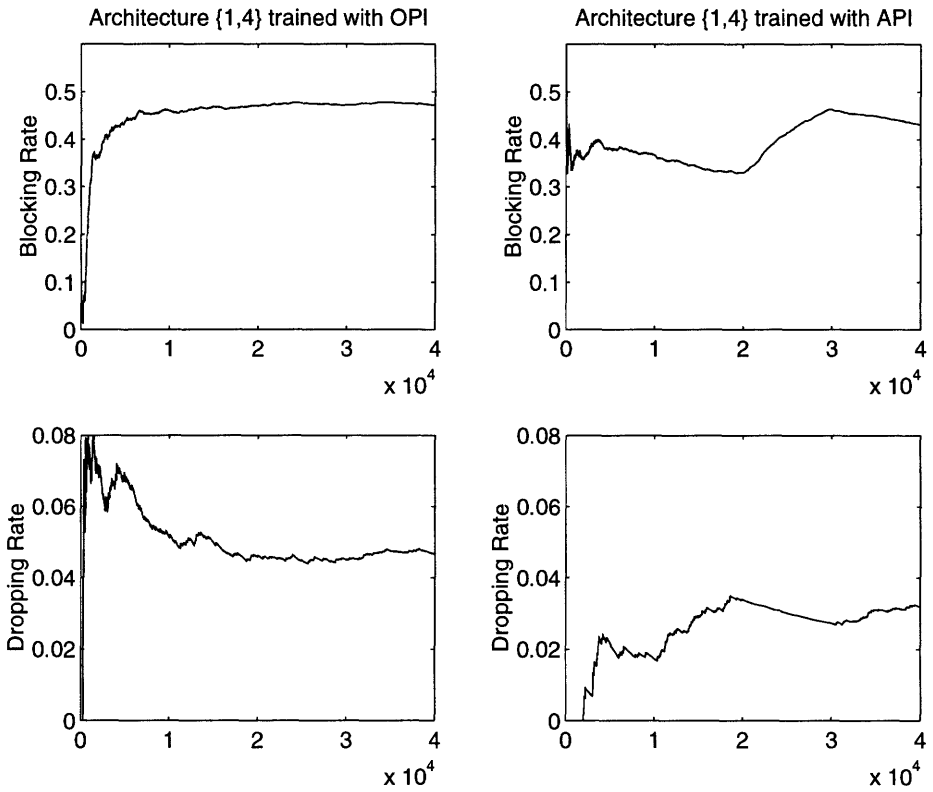


**Figure 4-17**: Performance of TD(0) in a system with non-uniform handoff probability and good initial weights.

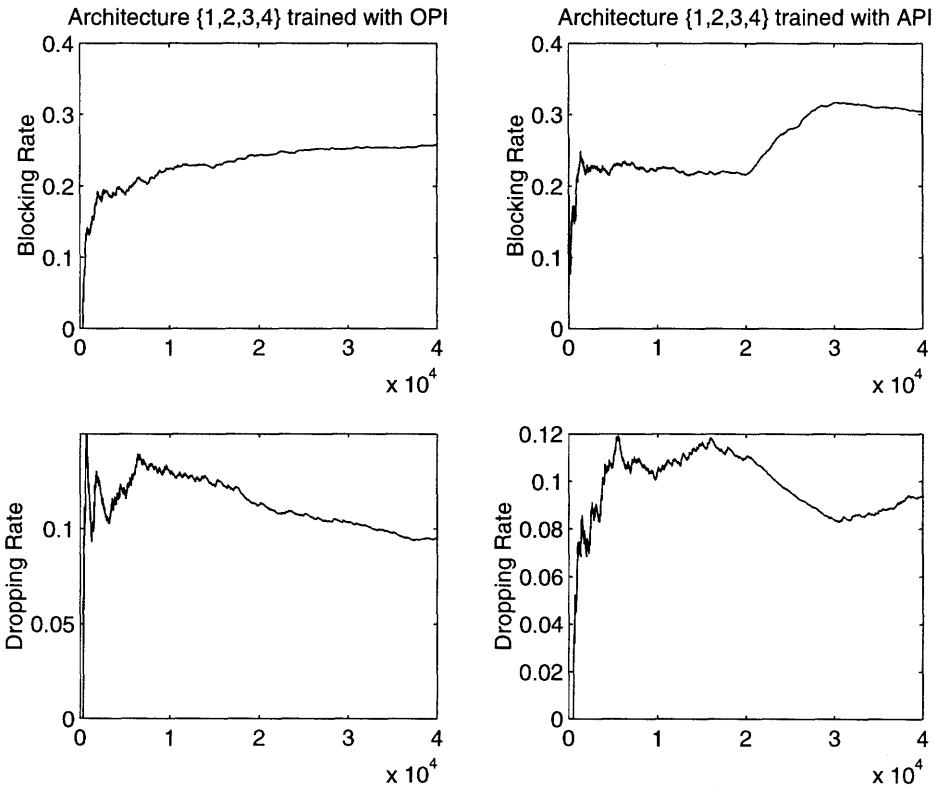


**Figure 4-18**: Performance of TD(0) in a system with non-uniform handoff probability and good initial weights.

# Chapter 5

# Conclusion

## 5.1 Discussion of Results

This thesis has presented both a success and a failure of the same methodology applied to two related but dissimilar problems. An exposition of the differences between the two problems will provide partial understanding of this gap in performance and perhaps illustrate some of the limitations of the methodology.

I present here a detailed analysis of the differences between the two problems and their possible relation to the results obtained. In the discussion that follows $\phi$ will denote a vector of features. The architecture $\tilde{J}$ is in the form of the inner product $w'\phi$, where $w$ is a parameter vector.

1. The first important difference between the two problems is the structure of the decision maker. In the channel allocation problem no instant costs are incurred therefore the decision is made by comparing the values of $w'\phi(x)$ for $x$ in some set of states. On the other hand, in the call admission problem the controller has to make a decision of the form $min\{c + \alpha w'\phi(x), \alpha w'\phi(\bar{x})\}$. The important difference here is that the absolute value of the weights in the first case is unimportant and only their relative magnitude

is significant, while in the call admission problem the absolute value of the weights is critical. To illustrate this claim suppose that there is only one feature in the architecture (the free channels for example). In the channel allocation problem every negative value of the weight will result to the same decision (the maximum availability heuristic). On the other hand, in the call admission problem a small $w$ will result in a policy that always rejects while a large value of $w$ will result in a policy that always accepts. Only a small range of values might give a non-trivial policy. Since training over a state space of that size involves a lot of noise, it is highly unlikely that the correct values for the weights (assuming that they exist) will be found.

2. For the channel allocation problem a good feature is already known, while in the call admission problem it is unclear what would be a good feature. I have experimented with heuristic policies that use estimates of the one-stage drop rate (the probability that the next event will be a drop) of the system but they were not very successful. Part of the reason why they were not successful is included in the following items in this list.

3. It appears that the call admission problem good heuristic policies must look ahead several steps into the future instead than just one step. The reason is that while in the channel allocation problem there is only one interesting type of events (call arrivals), in the call admission problem there are two competing types of events. Making decisions just by looking at next stage information cannot result in a good policy. This is the reason that channel reserving architectures perform so well. On the other hand, in the channel allocation problem, one stage lookahead is sufficient to obtain a good assignment strategy.

4. Finally, the call admission problem is more complicated in structure in that it involves a tradeoff. In the channel allocation problem the sole objective is to minimize the blocking rate. The call admission problem has to compromise the conflicting objectives of minimizing the blocking rate and the dropping rate. It is possible that linear architectures cannot capture this behavior unless perhaps they involve a feature that explicitly accounts for this tradeoff.

## 5.2 Extensions and Possible Improvements

### 5.2.1 Extensions

It was mentioned in chapter 2 that several cellular network systems use signal strength information to obtain better policies via the use of new call or handoff queuing. The issue of queuing can be addressed by the methods described in this thesis. In addition to the channel allocation and admission control decisions, a control for choosing a queued call to dequeue is introduced. In fact, I have addressed the problem of handoff queuing and I have extended the model and the simulator to handle these decisions. Unfortunately the simulation results were unsuccessful and are not presented in this thesis.

Another interesting extension of the problem is to consider different classes of users each with different service rate and different rewards. In this more general context handoff requests can be consider as new user arrivals of a different class. A successful application of the neuro-dynamic programming approach to admission control for multiple user classes in the context of routing in integrated service networks can be found in [16].

### 5.2.2 Possible Improvements

This research is by no means exhaustive. A particular class of methods was implemented and evaluated. There are other possible courses one can take that might lead to success. Here is a list of suggested variations.

- Using combinations of the suggested features with a neural network architecture. While a neural network with many parameters and layers is harder to train, the architectures obtained are much more powerful.
- The cost-to-go of complicated threshold policies has significant nonlinearities (see for example figure 4-4). As a result it might be hard for linear architectures to capture threshold policies. An alternative is to use an action network. Instead of constructing

an approximation of the cost to go, an approximation of the the policy is directly given by a function $\tilde{\mu}(x, r)$. Then policies are directly approximated by solving incrementally the least squares problem $\min_r \sum_{x \in \tilde{S}} (\tilde{\mu}(x, r) - \mu(x))^2$, where $\tilde{S}$ is the subset of states visited in a trajectory. Action networks as discussed in [3].

- It has been mentioned that the attempt solve both the admission control and channel allocation problem with the same architecture were unsuccessful. The same was true for the call admission problem with queues. The problem is probably that the features that are good for one type of decisions confuse the decision making for another type of decisions. An idea to get around this problem would be to create a separate architecture for each type of decisions. Clearly this idea is inapplicable if the training is done using an approximation of the cost-to-go, because there is only one cost-to-go function. However, if the action network approach is used then is no reason why there shouldn't be a different action network for each type of decisions, each with different features that are specific to the decision type.
- Clearly the weakness of the architecture is one of the prevailing problems here. In fact $\tilde{J}$ may be a very poor approximation to the optimal policy for any choice of the parameters. As a result, the controller makes decisions by comparing very noisy quantities. An idea due to professor Bertsekas for getting around this problem is instead of learning $\tilde{J}(x)$, to have an architecture that learns a function $\tilde{R}(x, \bar{x})$ that tries to approximate the difference of the cost-to-go between two neighboring states. Another idea, also due to professor Bertsekas, is to train an architecture that will depend both on the state and the controls $\tilde{\Delta}(x, u, u')$ that approximates the relative advantage of using control $u$ in state $x$ instead of control $u'$. This corresponds to approximate advantage updating. Advantage updating is discussed in [3].

# Bibliography


[1] Dimitri P. Bertsekas, 1995. *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont MA.

[2] Dimitri P. Bertsekas and Segey Ioffe, 1996. *Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming* LIDS-P-2349

[3] Dimitri P. Bertsekas, 1996. *Neuro-Dynamic Programming*, Athena Scientific, Belmont MA.

[4] D. C. Cox and D. O. reudink, *Dynamic Channel Assignment in Two Dimension Large-Scale Mobile Radio Systems*, Bell Sys. Tech. J., vol. COM-20, 1972, pp. 1611-28.

[5] Daigle J. N., R. Guerin, and N. Jain 1992. *Comments on "Queuing-Blocking System with Two-Arrival Streams and Guard Channel."*

[6] Daigle J. N., and Jain N. 1992. *A Queuing System with Two Arrival Streams and Reserved Servers with Application to Cellular Telephone.*

[7] J. S. Engel and M. Peritsky, *Statistical Optimum Dynamic Server Assignment in Systems with Interfering Servers*, IEEE Transactions on Vehicular Technology, vol. VT-22, 1973, pp. 203-9.

[8] Said M. Elnoubi, Rajendra Singh, Someshwar C. Gupta. *A New Frequency Channel Assignment Algorithm in High Capacity Mobile Communication Systems*, IEEE Transactions on Vehicular Technology, vol. VT-31, no. 3, 1982.

[9] Romano Fantacci, *A Dynamic Channel Allocation Technique Based on Hopfield Neural Netowrks* IEEE Transactions on Vehicular Technology, vol. 45 No.1 February 1996.

[10] Roch Guerin, *Queuing-Blocking System with Two Arrival Streams and Guard Channels*, IEEE Transactions on Communications, vol. 36, no.2, 1988.

[11] Simon Haykin, 1994 *Neural Networks. A Comprehensive Foundation* Mcmillan College Publishing Company, New York.

[12] D. Hong and S. S. Rappaport, *Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone Systems with Prioritized and Nonprioritizes Handoff Procedures*, IEEE Transactions on Vehicular Technology, vol. VT-35, no. 3, 1986.

[13] Se-Hyun and Dong-Wan Tcha *Prioritized Channel Assignment in a Cellular Radio Network* IEEE Trans. on Communications, Vol. 40, no. 7, July 1992

[14] T. J. Kahwa and N. Georganas, *A Hybrid Channel Assignment Scheme in Large Scale Cellular-Structured Mobile Communication Systems*, IEEE Transactions on Communications, vol. COM-26, 1978, pp. 432-38.

[15] I. Katzela and M. Naghshineh, *Channel Assignment Schemes for Cellular Mobile Telecommunication Systems: A Comprehensive Survey*, IEEE Personal Communications, June 1996.

[16] Peter Marbach, Oliver Mihatsch, Miriam Schulte, John N. Tsitsiklis, *Reinforcement Learning for Call Admission Control and Routing in Integrated Service Networks*, submitted to NIPS

[17] J. Sin and N. Georganas, *A Simulation Study of a Hybrid Channel Assignment scheme for Cellular Land-Mobile Radio Systems with Erlang-C Service*, IEEE Transactions on Communications, vol. COM-9, 1981, pp. 143-47.

[18] R. Singh, S. M. Eloubi, and C. Gupta, *A New Frequency Channel Assignment Algorithm in High Capacity Mobile Communications Systems*, IEEE Transactions on Vehicular Technology, vol. VT-31, 1982.

[19] Satinder Singh and Dimitri P. Bertsekas *Reinforcement Learning for Dynamic Channel Allocation in Cellular Telephone Systems.* NIPS 1996

[20] S. Tekinay and B. Jabbari, *A Measurement-Based Prioritization Scheme for Handovers in Mobile Cellular Networks*, IEEE Journal on Selected Areas in Communications, vol. 10, no. 8, 1992.

[21] Tesauro, G. J., 1995 *Temporal Difference Learning and TD-gammon* Communications of the ACM, Vol. 38, pp. 59-94.

[22] M. Zhang and T.S. Yum, *The Non-Uniform Pattern Allocation Algorithm for Cellular Mobile Systems*, IEEE Transactions on Vehicular Technology, vol. VT-40, 1991, pp.387-91.