

# Reinforcement Learning for Call Admission Control and Routing under Quality of Service Constraints in Multimedia Networks

HUI TONG, TIMOTHY X BROWN

{tongh, timxb}@colorado.edu

*Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309-0425*

**Abstract.** In this paper, we solve the call admission control and routing problem in multimedia networks via reinforcement learning (RL). The problem requires that network revenue be maximized while simultaneously meeting quality of service constraints that forbid entry into certain states and use of certain actions. The problem can be formulated as a constrained semi-Markov decision process. We show that RL provides a solution to this problem and is able to earn significantly higher revenues than alternative heuristics.

**Keywords:** Reinforcement learning, call admission control, routing, quality of service, multimedia networks.

## 1. Introduction

A number of researchers have recently explored the application of reinforcement learning (RL) to resource allocation and admission control problems in telecommunications, e.g., channel allocation in wireless systems, network routing, and admission control in telecommunication networks (Nie and Haykin, 1999, Singh, 1997, Boyan and Littman, 1994, Marbach, et al, 2000). This paper focuses on applications of the RL method to call admission control (CAC) and routing in broadband multimedia communication networks, such as Asynchronous Transfer Mode (ATM) networks and Multi-Protocol Label Switching (MPLS) networks. Broadband networks carry heterogeneous traffic types simultaneously on the same channel (so called multiplexing). The channels are packet-based so that customers can send at varying rates over time. Calls arrive and depart over time and the network can choose to accept or reject connection requests. If the new call is accepted, the network will choose an appropriate route to deliver the call from its source node to its destination node. The network provides Quality of Service (QoS) guarantees at the packet level, e.g., maximum probability of congestion, and at the call level, e.g. limits on call blocking probabilities. In return, the network collects revenue (payoff) from customers for calls that it accepts into the network. The network wants to find a CAC and routing policy that maximizes the long term revenue/utility and meets QoS constraints.

Maximizing revenue while meeting QoS constraints suggests a constrained semi-Markov decision process (SMDP), Mitra, et al (1998). The rapid growth in the number of states with problem complexity has led to RL approaches to the problem, Marbach and Tsitsiklis (1997), Marbach, et al (2000). However, these RL applications have ignored QoS criteria. This work draws on a closely related and

more fundamental problem of constrained optimization of (semi-)Markov decision processes, which has been studied by researchers from control theory, operation research and artificial intelligence communities, see e.g. Altman and Shwartz (1991), Feinberg (1994), Gabor, et al (1998).

Unlike model-based algorithms (e.g. linear programming in Mitra, et al, 1998), the RL algorithm used in this paper is a stochastic iterative algorithm, it does not require a priori knowledge of the state transition models (i.e., the state transition probabilities) associated with the underlying Markov chain, and thus can be used to solve real network problems with very large state spaces that cannot be handled by model-based algorithms, and can automatically adapt to real traffic conditions.

This work builds on earlier work of the authors (Brown, et al, 1999) in that it provides a more general framework for studying the CAC and routing problem, under QoS constraints. It also provides more detailed information and proofs for the RL algorithms used in the study which were not reported in Tong and Brown (2000).

Section 2 describes the problem model used in this study. Section 3 formulates the CAC problem as a SMDP, and gives a RL algorithm that solves the SMDP. Section 4 considers QoS constraints in more details. Simulations for CAC on a single link system is presented in Section 5. Combined CAC and network routing is studied in Section 6, with simulation results for a 4-node, 12-link network. Section 7 concludes the paper.

## 2. Problem description

This section describes the CAC problem for a single-link communication system. There is a substantial literature on CAC in one link multiservice networks, e.g. Marbach and Tsitsiklis (1997), Mitra, et al (1998), and references in Dziong and Mason (1994). The single link case is significant since it is the basic building block for larger networks and, as shown in Section 6 of this paper, combined CAC and routing for a multi-link network system can be decomposed into single link processes. We thus first focus on a single-link system.

Users attempt to access the link over time and the network immediately chooses to accept or reject the call. If accepted, the call generates traffic in terms of bandwidth as a function of time. At some later time, the call terminates and departs from the network. For each call accepted, the network receives an immediate revenue payment. The network measures QoS metrics such as transmission delays, packet loss ratios, or call rejection probabilities for each service class, and compares them against the guarantees given to the calls.

The problem is described by the call arrival, traffic, and departure processes; the revenue payments; QoS metrics; QoS constraints; and network model. To be concrete, we describe the choices used in the later examples. Calls are divided into discrete classes indexed by  $i, i = 1, 2, \dots, I$ . The calls are generated via independent Poisson arrival processes (arrival rate  $\lambda_i$ ) and have exponential holding times (mean holding time  $1/\mu_i$ ). Within a call the bandwidth is an ON/OFF process where the traffic is either ON, generating packets at rate  $r_i$ , or OFF at rate zero with mean

holding times  $1/\nu_i^{\text{ON}}$  and  $1/\nu_i^{\text{OFF}}$ . The ON/OFF holding times are not restricted to be exponentially distributed, this allows us to treat more general CAC problems with long-range dependence traffic (also referred to as self-similar or chaotic traffic, Leland, et al, 1993). When a class  $i$  call is admitted, the system collects a fixed amount of revenue  $\eta_i \in [0, \infty)$ , which can be interpreted as the average reward for carrying the  $i$ -th class call (Dziong and Mason, 1994). The network element connects to the network with a fixed bandwidth  $B$ . The total bandwidth used by accepted calls varies over time.

One important packet-level QoS metric is the fraction of time that the total bandwidth exceeds the network bandwidth and causes packet losses, i.e., the congestion probability,  $p_i, 1 \leq i \leq I$ . We choose the packet-level QoS guarantee to be the upper limit of congestion probability,  $p_i^*$ , which we denote the *Capacity constraint*. In previous works (e.g. Carlstrom and Nordstrom, 1997, Marbach, et al, 2000) each call had a constant bandwidth over time so that the effect on QoS was predictable. Variable rate traffic is safely approximated by assuming that it always transmits at its maximum or peak rate. This *peak rate allocation* under-utilizes the network; in some cases by orders of magnitude less than what is possible. Network efficiency can be improved by *statistical multiplexing*: Statistically, bursty sources are unlikely to all simultaneously communicate at their peak rates. Thus it is possible to carry more bursty or variable rate traffic than would be possible by allocating capacity according to peak rate requirements, while maintaining service quality. Stochastic traffic rates in real traffic, the desire for high network utilization/revenue, and the resulting potential for QoS violations characterize the problem in this study.

Another important QoS metric is the call-level blocking probability. When offered traffic from each class must be cut back to meet the capacity constraint, it is important to do so fairly, which we denote the *Fairness constraint*. Fairness can be defined in a number of different ways, but one intuitive notion is that calls from every class are entitled to the same admission probability, or equivalently, the same rejection probability, (Dziong and Mason, 1994). This will be more precisely defined in Section 4.

Ultimately our goal is to find a policy,  $\pi$ , that for every system state,  $s$ , chooses the correct control action,  $a$ , so that we maximize revenue subject to the QoS constraints. Formally, we consider the following problem of finding the CAC policy,  $\pi$ , that

$$\text{maximizes } J_0(\pi) \tag{1}$$

$$\text{subject to } J_j(\pi) \leq l_j, \quad j = 1, \dots, K, \tag{2}$$

$$\pi \in \{\text{set of all policies}\} \tag{3}$$

where  $K$  is the number of QoS constraints,  $l_j, j = 1, \dots, K$  are real numbers that characterize the QoS constraints,  $J_0(\pi)$  characterizes the average network revenue under policy  $\pi$ , and  $J_j(\pi), j = 1, \dots, K$  characterize the QoS under policy  $\pi$ . We consider objectives of the form

$$J_j(\pi) = \lim_{N \rightarrow \infty} \frac{\sum_{n=0}^{N-1} c_j(s_n, a_n)}{\sum_{n=0}^{N-1} \tau(s_n, a_n)} \tag{4}$$

for  $j = 0, 1, \dots, K$ . Action  $a_n$  is chosen at state  $s_n$  according to the policy  $\pi$ ,  $c_j(s_n, a_n)$  are the reward functions associated with revenue (for  $j = 0$ ) and QoS metrics (for  $j = 1, \dots, K$ ), and are assumed to be bounded.  $\tau(s_n, a_n)$  are the average sojourn times at state  $s_n$  under action  $a_n$ , while  $n$  indexes the  $n$ -th decision time point. (A sojourn time is the time from one transition to the next, when action  $a_n$  is applied at  $s_n$ . In the problem studied here the sojourn times are exponentially distributed.)

### 3. Semi-Markov decision processes and reinforcement learning

The following sections develop the components to the problem and finish by justifying a particular method suitable for the CAC problem.

#### 3.1. States and actions

This section develops the state action model and a reduced state space representation suitable for the CAC problem.

The CAC problem can be formulated as a semi-Markov decision process (SMDP) in which state transitions and control selections take place at discrete times, but the time from one transition to the next is a continuous random variable. At any given point of time, the system is in a particular configuration,  $\mathbf{x}$ , defined by the numbers of each type of ongoing calls, and  $\mathbf{y}$ , the numbers of calls in the ON state of each type if ON/OFF distributions are exponential. If ON/OFF distributions are not exponential, the  $\mathbf{y}$  would have to include, in addition, the elapsed ON/OFF holding time for each call in progress. At random times an event  $\mathbf{e}$  can occur (only one event can occur at any time instant), where  $\mathbf{e}$  is an  $I$ -vector indicating for class  $i$  either a call arrival, a call termination, a call being turned ON, or a call being turned OFF event,  $1 \leq i \leq I$ . The configuration and event together determine the state of the system,  $s = (\mathbf{x}, \mathbf{y}, \mathbf{e})$ . Clearly,  $s$  is at least a  $3I$ -dimensional vector for an  $I$ -class system. Since the number of possible choices for  $\mathbf{e}$  is in general small compared to those for  $\mathbf{x}, \mathbf{y}$ , the size of the state space is dominated by the configuration part of the state. It can be shown that using the *nearly complete decomposability* approximation we can reduce the state descriptor into the form of  $s = (\mathbf{x}, \mathbf{e})$ , where  $e_i$  stands for a call arrival or departure event of class  $i$ . Let  $\mathbf{x} = (x_1, x_2, \dots, x_I)$  be the configuration,  $\mathbf{e}_i$  denote the  $I$ -vector whose elements are equal to zero except the  $i$ th element, whose value is unity. Then the states associated with a class  $i$  call arrival are  $s = (\mathbf{x}, \mathbf{e}_i)$ , and the states associated with a class  $i$  call departure are  $s = (\mathbf{x}, -\mathbf{e}_i)$ ,  $1 \leq i \leq I$ . This reduction, by ignoring the number of calls in the ON state and the events of a call being turned ON or OFF, gives us enough accuracy for the CAC problem, as shown experimentally by Mitra, et al (1998).

Here we give two reasons for this simplification. First, the moment a call turns ON or OFF is not a decision point for the admission controller, and therefore no action needs to be taken and no reward will be incurred. Theorem 2 and its Corollary in the Appendix show that ignoring the events of a call being turned ON or OFF is

valid. Section 3.4 provides further discussions on similar simplifications. Second, it is intuitively clear that this simplification is a good approximation when the ON/OFF process (indicated by  $\mathbf{y}$ ) reaches equilibrium in a time faster than the call arrival/departure process (indicated by  $\mathbf{x}$ ). Hence, when making a call admission decision, the number of calls of each class in progress is important, but the number of calls of each class in the ON state is not, because these quantities oscillate rapidly relative to call arrivals and departures. If we view the dropping off of  $\mathbf{y}$  as state aggregation, and reasonably assume that for a fixed  $\mathbf{x}$ , the Q-values do not change much for different  $\mathbf{y}$ , then the discussions in Section 3.6 further justify this state reduction.

We note that  $\mathbf{y}$  *does* affect the congestion probability. Assuming the  $\mathbf{y}$  process reaches equilibrium and corresponds to fixed  $\mathbf{x}$ , and assuming source independence, the probability for the configuration  $(\mathbf{x}, \mathbf{y})$  is given by the binomial distribution

$$\psi(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^I \binom{x_i}{y_i} b_i^{y_i} (1 - b_i)^{x_i - y_i} \quad (5)$$

where  $b_i$  is the fraction of time a class  $i$  call spends in the ON state,

$$b_i = (\nu_i^{\text{ON}})^{-1} / [(\nu_i^{\text{ON}})^{-1} + (\nu_i^{\text{OFF}})^{-1}].$$

The average congestion probability for class  $i$  with fixed  $\mathbf{x}$  is thus

$$p_i(\mathbf{x}) = \sum_{y_1=0}^{x_1} \cdots \sum_{y_I=0}^{x_I} \left\{ \psi(\mathbf{x}, \mathbf{y}) \mathbf{1} \left\{ \mathbf{y} : \sum_j y_j r_j - B > 0 \right\} \mathbf{1} \{ \mathbf{y} : y_i r_i > 0 \} \right\} \quad (6)$$

where  $\mathbf{1}\{\cdot\}$  is the indicator function. So the average congestion probability depends only on  $\mathbf{x}$ .

Capacity constraints associated with (6) are *conservative* in that the set

$$C_c = \{ \mathbf{x} : p_i(\mathbf{x}) \leq p_i^*, 1 \leq i \leq I \} \quad (7)$$

is the set of  $\mathbf{x}$  such that the long run average packet-level QoS constraints are always satisfied, and we will never go into any state for any period of time where the capacity constraint will be violated if we stay there *forever*. The set  $C_c$  uniquely determines a state space  $S$ : for any  $i$ ,  $(\mathbf{x}, \mathbf{e}_i) \in S$  if and only if  $\mathbf{x} \in C_c$ ; and  $(\mathbf{x}, -\mathbf{e}_i) \in S$  if and only if  $x_i \geq 1$  and  $\mathbf{x} \in C_c$ . Mitra, et al (1998) considers a more *aggressive* approach to the packet-level QoS constraints, that averages across all the allowable configurations  $\mathbf{x}$ . Let  $T = \sum_{\mathbf{x} \in C_a} T(\mathbf{x})$  be the total system time,  $T(\mathbf{x})$  be the portion of  $T$  the system spends at  $\mathbf{x}$ , and  $C_a$  is the set of the allowable configurations  $\mathbf{x}$  such that

$$p_i = \lim_{T \rightarrow \infty} \frac{\sum_{\mathbf{x} \in C_a} p_i(\mathbf{x}) T(\mathbf{x})}{T} \quad (8)$$

are less than or equal to target  $p_i^*$ , for all  $i = 1, \dots, I$ . In some occasions, to emphasize the dependence of  $C_c$  and  $C_a$  on  $\mathbf{p}^* = (p_1^*, \dots, p_I^*)$ , we also write  $C_c(\mathbf{p}^*)$  and  $C_a(\mathbf{p}^*)$ .

In summary, we choose the state descriptor to be  $s = (\mathbf{x}, \pm \mathbf{e}_i)$ , where  $x_i$  is the number of class  $i$  calls in progress, and  $\mathbf{e}_i$  stands for a new class  $i$  call arrival,  $-\mathbf{e}_i$  for a class  $i$  call departure,  $1 \leq i \leq I$ .

When an event occurs, the learner has to choose an action feasible for that event. The action set is  $A(s) = \{0 = \text{reject}, 1 = \text{accept}\}$  upon a new call arrival. Call terminations are not decision points, so no action needs to be taken. Symbolically, at such states  $A(s) = \{-1 = \text{no action due to call departures}\}$ . Note that the actions available at state  $s$ ,  $A(s)$ , in general depend on  $s$ . For example, if adding a new call at a state  $s$  will violate the capacity constraint, then the action set at that state should be constrained to  $A(s) = \{0\}$ . At some subsequent random time another event occurs, and this cycle repeats. The network only earns revenue for accepted calls and the revenue depends on the call types:

$$c_0(s, a) = \begin{cases} \eta_i, & \text{if } \mathbf{e} = \mathbf{e}_i \text{ and } a = 1 \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

The task of the learner is to determine a policy for accepting calls given  $s$ , that maximizes the long-run average revenue, over an infinite horizon while meeting the QoS requirements. For CAC, the system constitutes a finite state space  $S = \{(\mathbf{x}, \mathbf{e})\}$ , (due to the capacity constraint), finite action space  $A = \{-1, 0, 1\}$ , semi-Markov decision process.

### 3.2. Transition probabilities

This section considers the probability model and concludes that for large state spaces, classical approaches based on the transition probability model are not feasible. Theoretically, a state transition probability  $p(s, a, s')$  — the probability of going from state  $s$  under action  $a$  to next state  $s'$  — can be derived (Mitra, et al, 1998), which depends on the configuration  $\mathbf{x}$  and call arrival/departure rates. But exact system models are often infeasible for several important reasons. First, call arrival rates may depend not only on each call class, but also on the configuration  $\mathbf{x}$  (Dziong and Mason, 1994). Therefore, the call arrival rate for each class may not be a constant in general. Second, for any network of reasonable size, the state space is extremely large. As an example, a 4-node, 12-link network with 3 service types has more than  $10^{45}$  states (Marbach, et al, 2000). It is not even possible to explicitly list all the states. Finally, fixing a model before computing the optimal policy means that it will not be robust if the actual traffic condition departs from the assumed model.

For the above reasons, it is clear that for any practical system with large state space, it will be very difficult, if not impossible, to determine the exact transition model for the Markov chain before performing any model-based algorithm to compute the optimal policy. This is the main motivation for this study to apply model-free RL algorithms to solve CAC problems. In particular, the Q-learning algorithm (Watkins and Dayan, 1992) is used in this paper. Q-learning can be performed using the actual system as a simulator to obtain sample transitions. In addition, the fact that Q-learning is an *off-policy* learning method (Sutton and

Barto, 1998, Section 7.6), means that even with arbitrary exploration actions, the final policy learned by Q-learning will still be optimal, see Section 3.5 for more discussions.

Although we will not explicitly compute the transition probabilities, we make the following assumptions in this study:

*Assumption A1.* Calls arrive according to independent Poisson processes, call holding times are exponentially distributed.

Let  $\xi_{ss'}(a)$  be the continuous random inter-transition time from state  $s$  to state  $s'$  under action  $a$ , with probability distribution  $F_{ss'}(\xi|a)$ . We have

*Assumption A2.*  $0 < \tau(s, a) < \infty$ , where  $\tau(s, a)$  is the expectation of  $\xi_{ss'}(a)$ , for any  $s \in S, a \in A(s)$ ,

$$\tau(s, a) = \sum_{s' \in S} p(s, a, s') \int_0^\infty \xi dF_{ss'}(\xi|a) \quad (10)$$

In particular, there exists  $\delta > 0, \varepsilon > 0$ , such that

$$\sum_{s' \in S} p(s, a, s') F_{ss'}(\delta|a) \leq 1 - \varepsilon, \quad \forall s \in S, a \in A(s) \quad (11)$$

*Assumption A3. Unichain Condition:* For every stationary policy  $\pi$ , transition matrix  $(p(s, \pi(s), s'))_{\{s, s' \in S\}}$  determines a Markov chain on  $S$  with one ergodic class and a (possibly empty) set of transient states.

Assumption A1 guarantees the transition probabilities are well defined. Assumption A2 states that the number of transitions in a finite time interval is, almost surely, finite. A3 guarantees that except for some initial transient states, any state can reach any other state with non-zero probability.

### 3.3. Q-learning

This section develops the RL methodology used in this paper for the unconstrained maximization of revenue. The QoS constraints will be considered in Section 4.

We learn an optimal policy using Watkins' Q-learning algorithm. Given optimal Q-values,  $Q^*(s, a)$ , the policy  $\pi^*$  defined by

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

is optimal. In particular, (12) implies the following procedures. When a call arrives, the Q-value of accepting the call and the Q-value of rejecting the call is determined. If rejection has the higher value, we drop the call. Else, if acceptance has the higher value, we accept the call. Only one action (and Q-value) exists at a call departure.

To learn  $Q^*(s, a)$ , we update our value function as follows: on a transition from state  $s$  to  $s'$  under action  $a$  in time  $\xi_{ss'}(a)$ ,

$$Q_{k+1}(s, a) = (1 - \gamma_k(s, a))Q_k(s, a) + \gamma_k(s, a) \left( c_0(s, a) + e^{-\alpha\xi_{ss'}(a)} \max_{b \in A(s')} Q_k(s', b) \right) \quad (13)$$

where  $\gamma_k(s, a) \in (0, 1]$  is the step size or learning rate,  $k$  is an integer variable to index successive updates, and  $\alpha > 0$  is chosen to be sufficiently close to 0 so that the discounted problem is equivalent to the average reward problem (the Tauberian approximation, Gabor, et al, 1998). Note that (13) is a general description of the Q-learning algorithm, and this algorithm is well known to be the Robbins-Monro stochastic approximation method that solves the so-called Bellman optimality equation associated with the decision process. Let

$$(HQ)(s, a) = c_0(s, a) + \sum_{s' \in S} \left[ p(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a) \right] \max_{b \in A(s')} Q(s', b) \quad (14)$$

for all  $s$ , and  $a \in A(s)$ . Assumption A2 guarantees that  $H$  is a contraction mapping with contraction factor

$$\zeta \triangleq \max_{s, a \in A(s)} \sum_{s' \in S} \left[ p(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a) \right] < 1 \quad (15)$$

with respect to the maximum norm.

**THEOREM 1.** *Suppose that*

$$\sum_{k=0}^{\infty} \gamma_k(s, a) = \infty \quad (16)$$

$$\sum_{k=0}^{\infty} \gamma_k^2(s, a) < \infty \quad (17)$$

for all  $s \in S, a \in A(s)$ , and each state-action pair is updated an infinite number of times. Then,  $Q_k(s, a)$  converges with probability 1 to  $Q^*(s, a)$ , for every  $s$  and  $a$ .

**Proof:** See Bertsekas and Tsitsiklis (1996). ■

### 3.4. A simplified learning process

There is a practical issue concerning the implementation of Q-learning (13). From the above discussions, Q-learning needs to be executed at every state transition, including the transition caused by a call departure, at which the feasible action set

is  $A(s) = \{-1\}$ , and  $c_0(s, a = -1) = 0$ . Since there is only one action at states associated with call departures, it is not necessary to learn the optimal Q-values at these states to induce the optimal policy at these states.

Is it possible to avoid the updates of Q-values at departure states, and still get the same optimal policy? This will reduce the amount of computation and storage of Q-values significantly, since the state space is almost halved by dropping the call departure states. We note that the only interesting states at which decisions need to be made are those associated with call arrivals,  $\{s = (\mathbf{x}, \mathbf{e}_j)\}$ . So the decision point jumps from one arrival to the next arrival, where an interarrival period may contain zero, one, or more departures. Given  $s = (\mathbf{x}, \mathbf{e}_i)$ ,  $a$ , and  $s' = (\mathbf{x}', \mathbf{e}_j)$ , where  $\mathbf{e}_j$  is the *first* arrival after  $\mathbf{e}_i$ , for the cases with  $n > 0$  departures between two adjacent arrivals, by Chapman-Kolmogorov equations (Bertsekas and Gallager, 1992), the transition probability for the actual decision process is

$$p^{(n+1)}(s, a, s') = \sum_{\hat{s}} p(s, a, \hat{s}) p^{(n)}(\hat{s}, b = -1, s') \quad (18)$$

$\hat{s}$  is the intermediate state corresponds to a call departure. More generally, let  $S_1 = \{s : |A(s)| = 1 \text{ and } c(s, b) = 0, b \in A(s)\}$  be the set of the intermediate states,  $S_2 = S - S_1 = \{s : s \in S \text{ and } s \notin S_1\}$ , then  $p^{(n)}(s, a, s')$  is the joint conditional probability  $p(s', N = n | s, a)$ , where  $N$  is the random variable such that  $s_0 = s \in S_2$ ,

$$N = \min\{n > 0 : s_n \in S_2\} \quad (19)$$

It is shown in the Appendix that the same optimal policy can be obtained by only doing Q-learning at the states associated with call arrivals. This result is intuitive since the call departures are random disturbances that only affect the state transitions. Even though (18) further complicates the already intractable transition model for the SMDP, since Q-learning does not depend on the explicit model, the asymptotic convergence to the optimal policy follows.

### 3.5. Exploration

In order for Q-learning to perform well, all potentially important state-action pairs  $(s, a)$  must be explored. Specifically, the convergence theorem of Q-learning requires that all state-action pairs  $(s, a)$  are tried infinitely often. This section develops an exploration strategy suitable for the CAC problem.

A common way to try all state-action pairs in RL is, with a small probability  $\epsilon$ , a random action rather than the action recommended by RL is chosen at each decision point during training, the so-called  $\epsilon$ -random exploration. For the CAC problem considered in this paper, without exploration some states will be visited with probabilities several orders higher than some other states, and experiments have shown that  $\epsilon$ -random exploration is very unlikely to help in this situation. Therefore, after training some states will be visited many times while some other states will only be visited for a few times, and the resulting Q-value functions are far from converging, and an optimal policy cannot be expected in reasonable time.

To see this, the call arrival process can be modeled as truncation of  $I$  independent M/M/ $\infty$  queues system. The truncated system is the same as for the untruncated system, except all configurations  $\mathbf{x} = (x_1, x_2, \dots, x_I)$  for which capacity constraint is violated have been eliminated. The stationary distribution of this system, assuming the *greedy* policy (in this paper the greedy policy means the policy that always accepts a new call if the capacity constraint will not be violated by adding the call), is given by Bertsekas and Gallager (1992)

$$P(x_1, x_2, \dots, x_I) = \frac{\frac{\rho_1^{x_1}}{x_1!} \frac{\rho_2^{x_2}}{x_2!} \dots \frac{\rho_I^{x_I}}{x_I!}}{G} \quad (20)$$

$G$  is a normalization constant,

$$G = \sum_{(x_1, x_2, \dots, x_I) \in C} \frac{\rho_1^{x_1}}{x_1!} \frac{\rho_2^{x_2}}{x_2!} \dots \frac{\rho_I^{x_I}}{x_I!} \quad (21)$$

where  $\rho_i = \lambda_i/\mu_i$ , and  $C$  is the allowed set of configurations of the truncated system. As an example, consider the same experimental parameters shown in Table 1 of Section 5 below, except that to simplify the calculation of the allowable configuration set  $C$  of the truncated system, we use the peak-rate allocation, so  $C = \{(x_1, x_2) : x_1 \times 0.08 + x_2 \times 0.2 \leq 1\}$ . Using (20) and (21) we have, for the most visited state,  $P_{\max} = 0.2297$ , and for the least visited state,  $P_{\min} = 1.1641 \times 10^{-6}$ , which result in approximately five orders of difference in the stationary distribution of state-action pairs for this small system. It is shown in Szepesvari (1998) that the convergence rate of Q-learning can be approximated by

$$|Q_k(s, a) - Q^*(s, a)| \leq \frac{L}{k^{(1-\zeta)P_{\min}/P_{\max}}} \quad (22)$$

for some suitable constant  $L > 0$ , where  $k$  is the same index as in (13), and  $\zeta$  as defined in (15).

To overcome the slow convergence caused by the small value of  $P_{\min}/P_{\max}$  in the stationary distribution, a controlled exploration scheme is derived based on the facts that Q-learning is an off-policy learning method (Sutton and Barto, 1998), and that the state-action pair distribution is linearly dependent on the distribution of configurations (Tong, 1999). At each state during training where there are more than one feasible actions, with probability  $\epsilon$  the control action is chosen that leads to the least visited configuration. This  $\epsilon$ -directed heuristic effectively reduces the difference in the number of visits between state-action pairs, and significantly speeds up the convergence of the value functions. In terms of the Q-learning formula (13), action  $a$  is chosen according to the exploration scheme, and action  $b$  is chosen according to the current Q-value.

In real CAC applications, to minimize possible performance degradation caused by the use of exploration, the following two-step learning procedure can be used. The CAC controller is first trained off-line using recorded traffic data from recent history which is expected to be very close to the actual traffic seen by the controller when it is put on-line. Large value of  $\epsilon$  can be used for quick convergence. Once

the algorithm converges and the controller is put on-line,  $\epsilon$  is changed to a much smaller value to track any (typically slow and small) changes in the actual traffic condition.

### 3.6. Function approximation vs. lookup tables

Q-learning deals effectively with the *curse of modeling* (an explicit state transition model is not needed, and a simulator can be used instead). Another major difficulty with SMDP problems is the *curse of dimensionality* (the exponential state space explosion with the problem dimension). In the above treatment, we have assumed that the problem state space is kept small enough so that a *lookup table* can be used. Clearly, when the number of state-action pairs becomes large, lookup 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 is necessary.

In this paper, we choose a state aggregation approximation architecture. We consider the partition of the state space  $S$  into disjoint subsets  $S_1, S_2, \dots, S_M$  and introduce a  $M$ -dimensional parameter vector  $\phi$  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

$$\tilde{Q}(s, a, \phi) = \phi(m, a), \quad \text{if } s \in S_m \quad (23)$$

When the value of  $M$  is small, a lookup table can be used for the aggregated problem where we learn  $\phi(m, a)$  instead of  $Q(s, a)$ . In this case, it can be shown (Bertsekas and Tsitsiklis, 1996) 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, and we choose to avoid them here.

In CAC, state aggregation can be interpreted as a feature-based architecture whereby we assign a common value  $\phi(m, a)$  to all states  $s$ , given  $a$ , that share a common feature vector. For example, a feature vector may involve for each call class a three value indicator, that specifies whether the load of each call class is “high”, “medium”, or “low” in the system, instead of specifying precisely the number of ongoing calls of each class,  $\mathbf{x}$ . Since states with similar numbers of calls would be expected to have similar Q-values, the error bounds in Tong (1999) guarantee the performance loss due to state aggregation is small. Therefore, the state space can be greatly reduced, and lookup table can be used.

### 3.7. Summary

This section formulates the CAC problem as a SMDP, and justify the Q-learning approach to solving the CAC problem. It shows that we can simplify the problem by ignoring the details of the within call processes and not computing Q-values for states that have no decision. Standard  $\epsilon$ -random exploration policies will significantly slow down learning in this problem, so a simple  $\epsilon$ -directed exploration

strategy is introduced. Aggregation of states is shown to be a simplifying heuristic that follows readily from the problem structure. The next section develops a method for incorporating the constraints into this framework.

#### 4. Constraints

We restrict the maximization to policies that never violate QoS guarantees, (1)–(3). For general SMDP problems, the constrained optimal policy is a *randomized* stationary policy, which randomizes in at most  $K$  states for a problem with  $K$ –constraints (Feinberg, 1994). However, model-based linear programming algorithms have to be employed to derive such a policy, which is impractical for CAC where the number of states can be very large. Since randomizations are needed at only  $K$  states, which is usually much smaller than the total number of states, the non-randomized stationary policy learned by RL is often a good approximation to the constrained optimal policy (Gabor, et al, 1998).

In general SMDP, due to stochastic state transitions, meeting such constraints may not be possible (e.g. from any state no matter what actions are taken there is a possibility of entering restricted states). In admission control, service quality depends on the number of calls admitted into the system and adding calls is strictly controlled by the admission controller, so that meeting such QoS constraints is possible.

We consider two important classes of QoS constraints in CAC in an integrated service network. One is the *State-dependent* constraints, the other is the *Past-dependent* constraints. The (conservative) capacity constraint is an example of state-dependent constraints. State-dependent constraints are QoS intrinsic to a state. The congestion probability is a function solely of the number of calls in progress of the current state, [cf. (6)]. Past-dependent constraints depend on statistics over the past history. An example is the fairness criterion. Fairness depends on the statistics of the rejection ratios over the past history. We address these two constraints separately.

##### 4.1. Capacity constraint

For simplicity, we consider a total packet congestion probability upper bound,  $p^*$ . For the conservative approach, this means the set  $C_c(p^*) = \{\mathbf{x} : p(\mathbf{x}) \leq p^*\}$ , where [cf (6) and (7)]

$$p(\mathbf{x}) = \sum_{y_1=0}^{x_1} \cdots \sum_{y_I=0}^{x_I} \left\{ \psi(\mathbf{x}, \mathbf{y}) \mathbf{1} \left\{ \mathbf{y} : \sum_j y_j r_j - B > 0 \right\} \right\} \quad (24)$$

As stated, the conservative capacity constraint is an intrinsic property of a state and it only depends on the current state. This allows us to collect QoS statistics about each state and treat them in a principled way (e.g. computing confidence intervals on the estimates). The current state and action  $(s_n, a_n)$  uniquely determine

the next configuration,  $\mathbf{x}_{n+1}$ , and the projected congestion probability for the next state  $s_{n+1}$  is determined only by  $\mathbf{x}_{n+1}$ . Therefore, to forecast the impact of  $a_n$  at  $s_n$ , we need to evaluate if  $p(\mathbf{x}_{n+1})$ , the expected congestion probability, is greater or less than the constraint  $p^*$ . If an action will cause  $p(\mathbf{x}_{n+1}) > p^*$ , this action should be eliminated from the feasible action set  $A(s_n)$ . In CAC, if adding a new call will violate the capacity constraint, then the only feasible action is to reject the new call request.

When considering the aggressive capacity constraint, we need to determine the set  $C_a = C_a(p^*)$  of allowable configurations, defined implicitly from

$$\lim_{T \rightarrow \infty} \sum_{\mathbf{x} \in C_a} p(\mathbf{x}) \cdot \frac{T(\mathbf{x})}{T} \leq p^* \quad (25)$$

where  $T(\mathbf{x})$  is the total time the system spends at  $\mathbf{x}$ , and  $T = \sum_{\mathbf{x} \in C_a} T(\mathbf{x})$ . We note that the distribution  $T(\mathbf{x})/T$  depends on the control policy. Generalization to the case where different service types have different packet-level QoS requirements can be easily made.

To construct the aggressive set  $C_a(p^*)$ , we choose conservative loss rate  $p_c^* \geq p^*$ . We gradually decrease  $p_c^*$  from 1, and find a series of sets  $C_c(p_c^*)$  corresponding to the changing  $p_c^*$ . Clearly, the size of  $C_c(p_c^*)$  is non-increasing with the decrease of  $p_c^*$ , however, it must always contain  $C_c(p^*)$ . In practice, at some value  $p_{c0}^*$  *under the learned policy*, if the aggressive congestion probability will be sufficiently close to, and still less than the constraint  $p^*$ , then the search for  $C_a(p^*)$  can stop, and we choose the  $C_a(p^*) = C_c(p_{c0}^*)$  for the aggressive capacity constraint. In essence, we try to find a corresponding value of conservative threshold  $p_c^*$  to the aggressive threshold  $p^*$ , and construct  $C_a$  from the conservative approach. This way, the aggressive capacity constraint remains to be a state-dependent constraint, and as for the conservative capacity constraint, we can implement this constraint by constraining the action set at each state. Although  $C_a$  determined in the above way may not be the most aggressive one in term of the revenue maximization (1)–(3), the loss of optimality is small (Tong, 1999).

#### 4.2. Fairness constraint

Let  $R_i(s_n)$  be the measured rejection ratio for class  $i$  upon the  $n$ th call arrival (before the  $n$ th decision is made). For arbitrary constraints on  $R_i(s_n)$ , we may not be able to find a feasible policy. The fairness constraint involves comparisons of rejection ratios for all types of calls. We formulate the fairness constraints as

$$f(\mathbf{R}(s_n)) = \max_{1 \leq i \leq I} R_i(s_n) - \min_{1 \leq i \leq I} R_i(s_n) \leq l_d \quad (26)$$

where  $l_d$  is the maximum allowed rejection ratio discrepancy. A feasible policy exists by always rejecting all call types. The aggressive fairness constraint can be formulated as

$$\lim_{N \rightarrow \infty} \frac{\sum_{n=0}^{N-1} f(\mathbf{R}(s_{n+1})) \cdot \xi_{s_n s_{n+1}}(a)}{\sum_{n=0}^{N-1} \xi_{s_n s_{n+1}}(a)} \leq l_d \quad (27)$$

where  $\xi_{s_n s_{n+1}}(a)$  is the inter-transition duration from state  $s_n$  to  $s_{n+1}$  under action  $a$ . This formulation is a constrained SMDP problem (1)–(3) with  $K = 1$ , since the capacity constraint is implemented by constraining the feasible action set at each state as described in the preceding subsection.

To deal with the fairness constraint, we use the Lagrange multiplier framework studied in Beutler and Ross (1986). Since the fairness constraint is a past-dependent constraint (the vector  $\mathbf{R}(s_{n+1})$  depends on the rejection ratios over the past history), to fit into this framework, we need to include this history information into our state descriptor. The new state descriptor,  $\bar{s}$ , has the form

$$\bar{s} = (\mathbf{req}, \mathbf{rej}, \xi, s) \quad (28)$$

where the  $I$ -vector  $\mathbf{req}$  (resp.  $\mathbf{rej}$ ) denotes the total number of call requests (resp. rejections) from each class before the current call arrival,  $\xi$  is the time interval between the last and the current call request, and  $s = (\mathbf{x}, \mathbf{e})$  is the original state descriptor. We obtain a Markov chain by doing this expansion, however, the state space has been enlarged significantly. Specifically, due to the inclusion of  $\mathbf{req}$ ,  $\mathbf{rej}$ , and  $\xi$ , the state space is now infinite, and we must resort to some form of function approximation to solve the SMDP problem. Again, we use state aggregation as our approximation architecture, by quantizing the rejection ratios  $R_i = rej_i/req_i$  and  $\xi$ .

In terms of a Lagrange multiplier  $\omega$ , we consider the unconstrained optimization for the parametrized reward

$$c^\omega(\bar{s}, a, \bar{s}') = c_0(\bar{s}, a, \bar{s}') - \omega c_1(\bar{s}, a, \bar{s}') \quad (29)$$

where  $c_0(\bar{s}, a, \bar{s}') = c_0(s, a)$  is the original reward function associated with the revenue,  $c_1(\bar{s}, a, \bar{s}')$  is the cost function associated with the constraint

$$c_1(\bar{s}, a, \bar{s}') = f(\mathbf{R}(\bar{s}')) \cdot \xi_{\bar{s}\bar{s}'}(a) \quad (30)$$

i.e.,  $c_1$  equals the numerator term of (27).

If there exists a non-randomized policy  $\pi^{\omega^*}$  that solves the Bellman optimality equation associated with reward function (29), and in the mean time, achieves the equality in (27), then Beutler and Ross (1986) shows that  $\pi^{\omega^*}$  is the constrained optimal policy. In case such optimal policy does not exist, it is shown that the constrained optimality is achieved by randomization at only one state  $\bar{s}_0$ , between two non-randomized policies,  $\pi_1^{\omega^*}$  and  $\pi_2^{\omega^*}$ , that only differ from each other in  $\bar{s}_0$ , with  $\pi_1^{\omega^*}$  (resp.  $\pi_2^{\omega^*}$ ) slightly undershooting (resp. overshooting)  $l_d$ . Clearly, in case that the non-randomized constrained optimal policy does not exist,  $\pi_1^{\omega^*}$  is the next best non-randomized policy, and the loss of optimality is minimal. For the above reasons, and to avoid the complications of randomized policies, we concentrate on non-randomized policies in this study. A nice monotonicity property associated with  $\omega$  shown by Beutler and Ross (1985) facilitates the search for  $\omega^*$ .

### 4.3. Summary

This section shows how the constraints can be introduced to the problem, either by modulating the action space or modifying the reward function. While optimality requires a randomized policy, since the policy only needs to be randomized in two ( $K = 2$ ) states out of many states, we greatly simplify the search by restricting ourselves to deterministic policies.

## 5. Simulation results

The experiments use the following model. The total bandwidth is normalized to 1.0 unit of traffic per unit time. The target congestion probability is  $p^* = 10^{-3}$ . Two source types are considered with the properties shown in Table 1. The fairness constraint is that the average rejection ratio discrepancy for two service types should not differ more than  $l_d = 5\%$ . As noted before, all holding times are exponential.

Table 1. Experimental parameters

Parameter	Source Type	
	I	II
ON rate, $r$	0.08	0.2
Mean ON period, $1/\nu^{\text{ON}}$	5	5
Mean OFF period, $1/\nu^{\text{OFF}}$	15	45
Call arrival rate, $\lambda$	0.067	0.2
Call holding time, $1/\mu$	60	60
Immediate payoff, $\eta$	5	1

We first concentrate on the conservative approach to the capacity constraint. Since exploration is employed to ensure that all potentially important state-action pairs are tried, it naturally enables us to collect statistics that can be used to estimate QoS at these state-action pairs. It should be emphasized that a single visit to a state is not sufficient to determine the long run QoS metrics due to variability in the within call process. As the number of times that each state-action pair is visited increases, the estimated service quality becomes more and more accurate and, with confidence, we can gradually eliminate those state-action pairs that will violate QoS requirements. As a consequence, the value function is updated in a gradually correct subset of state-action space in the sense that QoS requirements are met for any action within this subspace. As stated in Section 4, the capacity constraint eliminates those state-action pairs that violate the congestion probability upper limit.

In the experiments, we use a simple way to eliminate state-action pairs with confidence. Since our target congestion probability is  $p < p^* = 10^{-3}$ , let  $T(\mathbf{x})$  be the total number of visits to the configuration  $\mathbf{x}$ , (counted as the number of time steps in the simulation), and  $w(\mathbf{x})$  be the number of these visits that were congested, then if  $\frac{w(\mathbf{x})}{T(\mathbf{x})} > 10^{-1}$  and  $T(\mathbf{x}) > 200$ , or if  $\frac{w(\mathbf{x})}{T(\mathbf{x})} > 10^{-2}$  and  $T(\mathbf{x}) > 2000$ , or if  $\frac{w(\mathbf{x})}{T(\mathbf{x})} > 10^{-3}$  and  $T(\mathbf{x}) > 20000$ , we conclude that  $(s, a)$  is not acceptable. These thresholds

provide close approximations to the confidence intervals in Brown (1997). A more sophisticated way to estimate  $p(\mathbf{x})$  is proposed in Tong and Brown (1998), where artificial neural networks are trained based on the maximum likelihood principle so that the neural network estimates of  $p(\mathbf{x})$  extrapolate well down to  $p^* = 10^{-8}$ . In simulations, the discount factor  $\alpha$  is chosen to be  $10^{-4}$ , learning rate  $\gamma = 0.01$ , and exploration probability  $\epsilon = 1$ . Initial Q-values for RL are artificially set such that Q-learning started with the greedy policy, i.e.,  $Q(s, \text{accept}) > Q(s, \text{reject})$ .

After training is completed, we apply a test data set to compare the policy obtained through RL with alternative heuristic policies. The final QoS measurements obtained at the end of the RL training while learning QoS are used for testing different policies. To test the RL policies, when there is a new call arrival, the algorithm first determines if accepting this call will violate QoS. If it will, the call is rejected, else the action is chosen according to  $a = \arg \max_{a \in A(s)} Q(s, a)$ , where  $A(s) = \{1=\text{accept}, 0=\text{reject}\}$ . For the QoS constraint we use three cases: Peak rate allocation; statistical multiplexing function learned on-line, denoted QoS learned; and statistical multiplexing function given a priori, denoted QoS given. We examine six different cases: (1) RL: QoS given. Since the statistical multiplexing function is given, only the RL policy needs to be learned; (2) RL: QoS learned, where both statistical multiplexing function and RL policy are learned at the same time; (3) RL: peak rate, where RL policy is learned assuming no statistical multiplexing; (4) A heuristic that only accepts calls from the most valuable class (type I), with statistical multiplexing (QoS given); (5) Greedy: QoS given, where the greedy policy is used with the given statistical multiplexing function; and (6) Greedy: peak rate, i.e., greedy policy with no statistical multiplexing.

From the results shown in Fig. 1, it is clear that simultaneous Q-learning and QoS learning converges correctly to the RL policy obtained by giving the QoS a priori and doing standard Q-learning only. We see significant gains (about 15%) due to statistical multiplexing in (6) vs (5), and (3) vs (1). The gains due to RL are about 25% in (6) vs (3), and (5) vs (2). Together they yield about 45% increase in revenue over conservative peak rate allocation in this example. It is also clear from the figure that the RL policies (1,2,3) perform better than the heuristic policies (4,5,6). Fig. 2 shows the rejection ratios for different policies. There are four pairs of bars in this figure for four different policies, the left bar of each pair is the rejection ratio for type I calls, and the right bar for type II calls. Clearly, RL policies accept much more type I (more valuable) calls than the heuristic policies.

Now we consider the aggressive approach to the capacity constraint. From the simulation, it is found that the value of  $p_{c0}^* = 3.9 \times 10^{-3}$  corresponds to the aggressive capacity constraint  $p^* = 10^{-3}$ . The acceptance regions (i.e.,  $C_a$  and  $C_c$ ) for both the aggressive and conservative approaches are shown in Fig. 3. The aggressive acceptance region is much larger than the conservative one. In the figure, the number of type II users starts at two due to insufficient measurement data (for the confidence level) in the region below that. Comparing Figs. 4 and 5 with Figs. 1 and 2, we can see that the aggressive approach earns significantly more revenue than the conservative approach, for both greedy policy and RL policy, note that the peak rate allocation earns the same total amount of rewards (un-normalized)

with both approaches. In Fig. 4, the Q-values are initialized so that the RL policy starts with the greedy policy.

In the above examples, the performance improvement due to RL is more significant than the improvement due to statistical multiplexing. Because no fairness constraint is imposed for this case, rejection ratios for the two types of calls differ significantly.

Our fairness constraint requires that the two rejection ratios cannot differ more than 5% on average. To test RL under the fairness constraint, we set the reward parameters for a type I call to be 1, and for a type II call to be 10, and keep other parameters in Table 1 unchanged. As stated before, we use feature-based state aggregation to cope with the difficulty of the large state space caused by the fairness constraint. Specifically, we learn  $Q(h(\bar{s}), a)$  instead of  $Q(\bar{s}, a)$ , where the feature  $h(\bar{s}) = ([R_1(\bar{s}) - R_2(\bar{s})], [\xi], s)$ ,  $[\cdot]$  denotes quantization. In the following experiment, the experienced rejection ratio discrepancy  $R_1(\bar{s}) - R_2(\bar{s})$  is quantized into 100 levels, and  $\xi$  is quantized into only 2 levels, with  $[\xi] = 1$  corresponding to  $\xi \leq 4$  (the approximate average inter-arrival time), and  $[\xi] = 2$  to  $\xi > 4$ . Although  $s$  is not aggregated in this experiment, for cases where  $s$  is more complicated, it is also possible to aggregate  $s$  into a simpler feature. Using binary search,  $\omega^*$  is found to be 80.0 in the simulation. The learned RL policy is compared with a greedy policy under the fairness constraint, which accepts all calls as long as the fairness constraint is met, otherwise, if the fairness constraint is violated, it only accepts calls from the class experiencing highest rejection ratio. The results are shown in Figs. 6 and 7. With fairness as a strong constraint on possible policies, we are forced to accept more less valuable calls, resulting in rejecting more call requests from the more valuable class due to the limited bandwidth resource, therefore, RL gain reduces in this case.

From Figs. 1, 4, and 6, we see that Q-learning converges quickly. The fact that the RL curves in these figures show oscillations is connected with the learning rates,  $\gamma_k(s, a)$  in (13). Specifically, in order for Q-learning to converge,  $\gamma_k(s, a)$  have to satisfy (16) and (17) for all  $(s, a)$ . But in the simulations, we used a small constant learning rate,  $\gamma = 0.01$ , i.e., condition (17) is not met. The reason that (17) is not adhered to is because typically, there is no prior knowledge as to how and when  $\gamma_k(s, a)$  should be decreased — once the learning rate becomes very small, the algorithm may stop making any noticeable progress, and the training process could become too long.

## 6. Combining CAC with network routing

In general, the issues of CAC and routing are closely related in a communication network. Combined CAC and routing can also be formulated as a SMDP. However, the exact characterization of network state would require the specification of the number of calls in progress from each class on each possible route in the network. Such a detailed specification of state is intractable for computation. By assuming statistical independence of the links in the network (Dziong, 1997, Krishnan, 1990), some form of decompositions of network routing process into single link processes

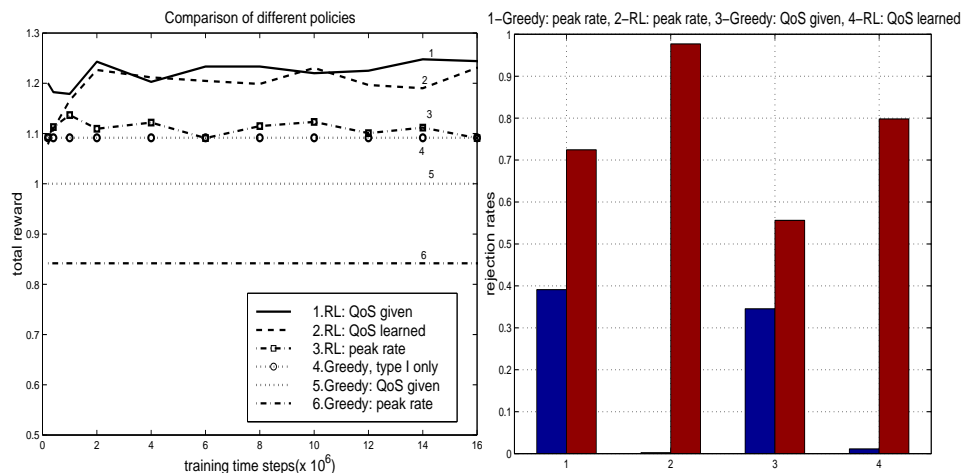


Figure 1. Comparison of total rewards of RL while learning QoS (capacity constraint only), RL with given QoS measurements, RL with peak rate, greedy policies and peak rate allocation, normalized by the greedy total reward. Uses the conservative capacity constraint.

Figure 2. Comparison of rejection rates for the policies learned in Fig. 1. For each pair of bars: left bar for type I, right bar for type II.

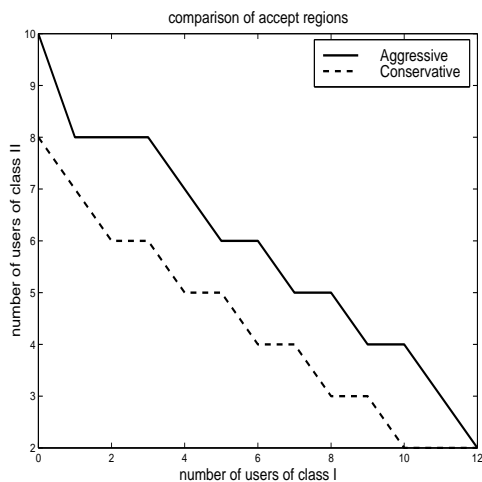


Figure 3. Comparison of acceptance regions for aggressive and conservative capacity constraints.

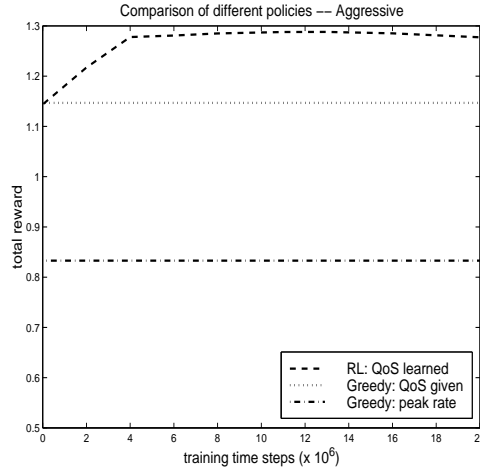


Figure 4. Fig. 1 with aggressive capacity constraint.

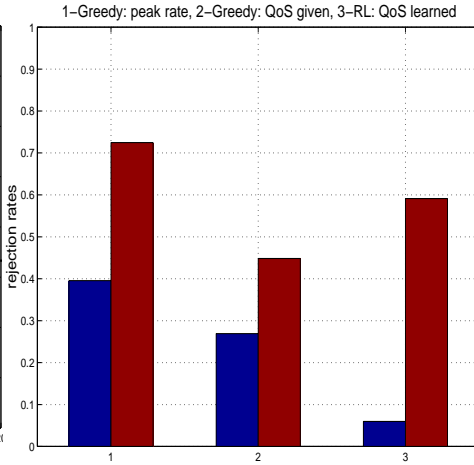


Figure 5. Comparison of rejection ratios for the policies learned in Fig. 4. For each pair of bars: left bar for type I, right bar for type II.

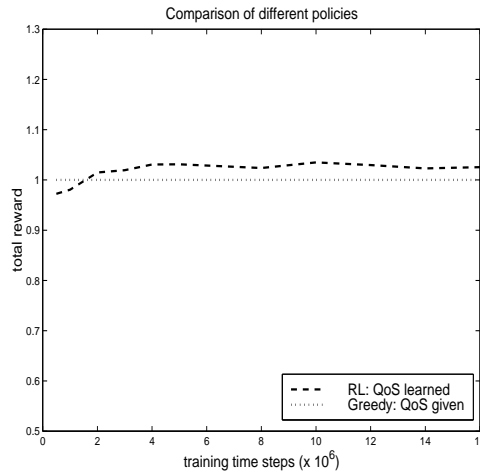


Figure 6. Comparison of total rewards obtained from RL policy and Greedy policy, when both capacity constraint and fairness constraint are imposed, normalized by the greedy total reward.

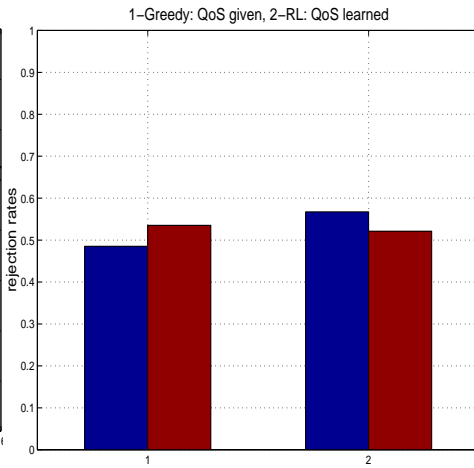


Figure 7. Comparison of rejection ratios with both capacity constraint and fairness constraint for the policies learned in Fig. 6. For each pair of bars: left bar for type I, right bar for type II.

is usually employed (Dziong and Mason, 1994, Marbach, et al, 2000). Based on the preceding results for single link CAC and the link state independence approximation, a new decomposition rule is derived in Tong (1999), it allows decentralized training and decision making for combined CAC and routing in a network, which also tries to maximize the network revenue. Due to the space limitation, we briefly describe the algorithm and present simulation results in this paper.

Let  $\mathcal{R}$  denote all the predefined routes in the network, the action space for the system is  $A = \{-1 = \text{no action due to call departures}, 0 = \text{reject}, r = \text{route the new call over route } r \in \mathcal{R}\}$ . Each link  $(i, j)$ , from node  $i$  to node  $j$ , keeps a separate Q-function,  $Q_{ij}(s^{ij}, r)$ , where  $s^{ij}$  is the link state variable. Whenever a new call of type  $k$  is routed over a route  $r$  which contains link  $(i, j)$ , the immediate reward associated with the link  $(i, j)$  is equal to  $c^{ij}$  satisfying

$$\sum_{(i,j) \in r} c^{ij} = \eta_k \quad (31)$$

For example,  $c^{ij} = \eta_k/|r|$ , where  $|r|$  is the number of links along the route  $r$ . Q-learning can be performed for each link similarly as in the single link case. At each arrival, we update the Q-value of link  $(i, j)$ , only if this arrival is *associated* with the link, meaning the new call can be potentially routed over link  $(i, j)$ . For a new call originated at node  $o$ , destined for node  $d$ , the decision is made at node  $o$  in the following way:

- i) Set  $g_0 = 0$ , and let  $A(s^{od})$  be the set of routes that can carry the call without violating QoS constraints.
- ii) Define the net gain  $g_r$  of routing the new call over route  $r \in A(s^{od})$  by

$$g_r = \sum_{(i,j) \in r} [Q_{ij}(s^{ij}, r) - Q_{ij}(s^{ij}, 0)]. \quad (32)$$

The admission and routing decision is

$$r^* = \arg \max_{r \in A(s^{od}) \cup \{0\}} g_r \quad (33)$$

**Decision Making:** If  $r^* = 0$ , then reject the call. Otherwise, accept the call and route it over route  $r^*$ .

That is, the routing policy chooses an action  $r^*$  that achieves maximum net gain  $g_{r^*}$ . The determination of the set  $A(s^{od})$  is out of the scope of this paper, we only mention that our approach of handling QoS in the single link case (see also Tong and Brown, 1998, Brown, 2001) can be extended to determine  $A(s^{od})$  in the network case by using QoS routing algorithms described by Lee, et al (1995), since the end-to-end QoS in a network is determined by the QoS of each link that consists the route, Pornavalai, et al (1997).

In the above approach, although the network state  $s$  is simplified into the link state  $s^{ij}$  for each link, the action space for each link is not simplified into  $\{0 = \text{reject}, 1 =$

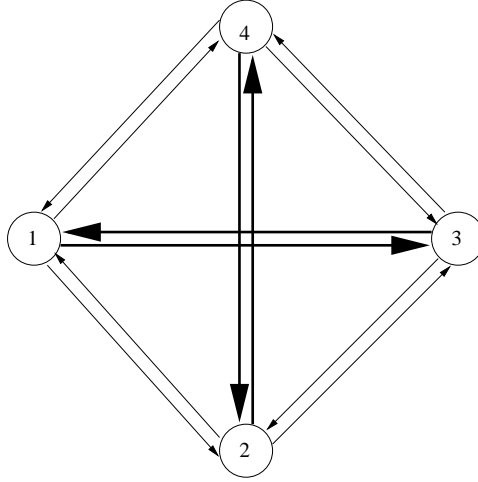


Figure 8. Network model.

accept}, as in Dziong and Mason (1994), Marbach, et al (2000). This is important since by doing so the link Q-functions can distinguish single-link calls from multi-link calls, and avoid accepting too many multi-link calls, and block single-link calls that may bring the same amount of revenue while using less network resources (Tong, 1999).

Table 2. Experimental parameters

Parameter	Source Type		
	I	II	III
On rate, $r$	0.05	0.1	0.2
Call arrival rate, $\lambda$	0.06	0.125	0.045
Call holding time, $1/\mu$	200	240	154
Immediate payoff, $\eta$	1	1	8

We present simulation results obtained for the case of a network consisting of 4 nodes and 12 unidirectional links. There are two different classes of links with a total bandwidth of 1.5 and 2 units, respectively (indicated by thin and thick arrows in Fig. 8). We assume three different source types, whose parameters are given in Table 2. Call arrivals at each node are independent Poisson processes with mean  $\lambda$ , the destination node is randomly selected among the other three nodes. For each source and destination node pair, the list of possible routes consists of three entries: the direct path and two alternative 2-hop routes. To emphasize the effect of RL,

we only consider the capacity constraint and assume peak rate allocation on each link in the simulations.

We use feature-based state aggregation to approximate the Q-values for each link, where we learn  $Q(h(s), r)$  instead of  $Q(s, r)$ , and  $h(s) = ([\mathbf{x}]_s, \mathbf{e})$ , i.e., the numbers of ongoing calls from each type are aggregated into eight levels.

The policy obtained through RL is compared with a commonly used heuristic policy that gives the direct path priority. When the direct path reaches its capacity, the heuristic will try the 2-hop routes, and find one that does not violate the capacity constraint. If no such route exists, the call is rejected.

The results are given in Fig. 9 (Total revenue), Fig. 10 (Call rejection ratios), and Fig. 11 (Routing behavior). The results show that the RL policy increases the total revenue by almost 40% compare to the commonly used heuristic routing policy. The RL policy accepts almost all calls from the most valuable type (type III), and uses two-hop alternative routes only for a small portion of type III calls, while routes the less valuable type I and II calls almost exclusively over the direct routes. This indicates that RL policy allows more efficient use of network resources.

## 7. Conclusion

This paper formulates the CAC and routing problem as a constrained SMDP, and provides a RL algorithm for computing the optimal control policy. We incorporate two important classes of QoS constraints, state-dependent and past-dependent constraints, into a RL solution to maximize a network's revenue. The formulation is quite general and has been applied to the capacity and fairness constraints. The approach was experimented with on a single link as well as a network problem, and we showed significant improvement even for simple examples.

Future work includes studying other function approximators, such as neural networks, to approximate Q-value functions.

## Acknowledgments

The authors would like to acknowledge the effort of anonymous reviewers and Leslie Pack Kaelbling whose remarks have influenced the final shape of this paper. This work was funded by NSF CAREER Award NCR-9624791.

## Appendix

### Proof for the simplified learning process

The following Theorem shows that we can avoid learning Q-values at state transitions corresponding to calls being turned ON or OFF (Section 3.1), and call departures (Section 3.4). Let  $J^\alpha(s) \triangleq \max_{a \in A(s)} Q^*(s, a)$  when the discount rate is  $\alpha$ , let  $S_1$  and  $S_2$  be as defined in Section 3.4. Since in our problem calls arrive according to independent Poisson processes and the holding times are exponentially distributed, the state transition processes are memoryless.

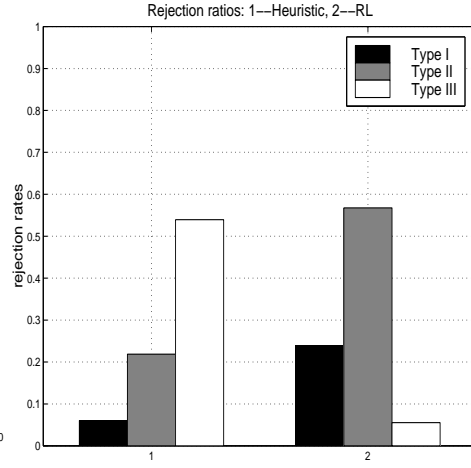
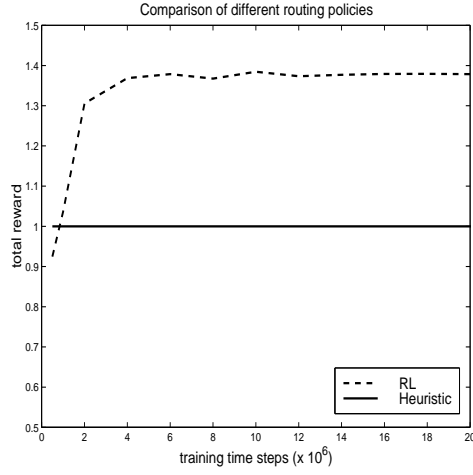


Figure 9. Comparison of the total rewards for the 4-node network, normalized by the Heuristic total reward.

Figure 10. Comparison of the rejection ratios for the policies in Fig. 9.

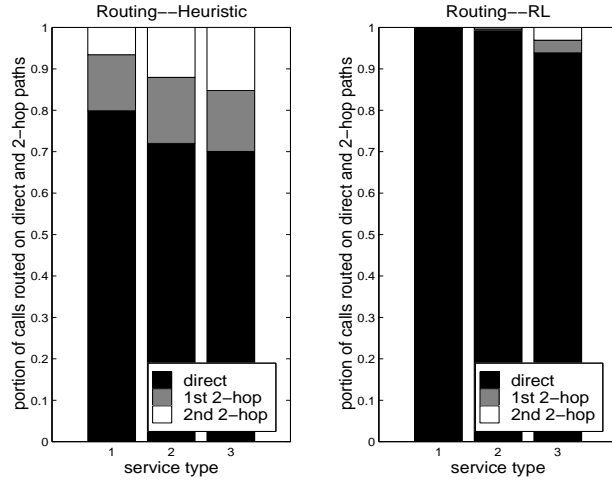


Figure 11. Comparison of the routing behavior for the policies in Fig. 9.

**THEOREM 2.** *Assume that from each  $s \in S_2$ ,  $a \in A(s)$ , it takes at most  $m_{sa}$  steps to go to a state  $\hat{s}' \in S_1$ , such that  $\sum_{s' \in S_2} p(\hat{s}', b, s') = 1$ , where all the states between  $s$  and  $\hat{s}'$  are in  $S_1$ , and  $0 < m_{sa} < \infty$ . Then the optimal stationary policy for the modified decision process by only considering states in  $S_2$  is also optimal for the original decision process.*

**Proof:** The optimal policy for the original problem is

$$\pi_0^\alpha(s) = \arg \max_{a \in A(s)} \left\{ c(s, a) + \sum_{s' \in S} \left[ p(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a) \right] J_0^\alpha(s') \right\} \quad (\text{A.1})$$

for all  $s \in S$ . Since there is only one feasible action in  $A(s)$  for  $s \in S_1$ , the optimal policy,  $\pi^\alpha$ , for the modified decision process is,

$$\pi^\alpha(s) = \pi_0^\alpha(s), \quad \forall s \in S_1 \quad (\text{A.2})$$

For each  $s \in S_2$ ,  $n$  in (18) can be at most  $m_{sa}$  by assumption.

$$\begin{aligned} \pi^\alpha(s) = \arg \max_{a \in A(s)} \left\{ c(s, a) + \sum_{s' \in S_2} \left[ p(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a, N = 1) \right. \right. \\ \left. \left. + p^{(2)}(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a, N = 2) + \dots \right. \right. \\ \left. \left. + p^{(m_{sa}+1)}(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a, N = m_{sa} + 1) \right] J^\alpha(s') \right\} \quad (\text{A.3}) \end{aligned}$$

where  $s'$  is the *first* state after  $s$  that is in  $S_2$ , and  $N$  is the random variable defined in (19). Define

$$D^{(n)}(s, a, s') \triangleq \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a, N = n)$$

Since  $m_{sa}$  is finite, without loss of generality, assume  $m_{sa} = 2$  (for other values of  $m_{sa}$ , the proof goes through in an identical manner). The summation term in (A.3) becomes

$$\begin{aligned} & \sum_{s' \in S_2} \left[ p(s, a, s') D(s, a, s') + p^{(2)}(s, a, s') D^{(2)}(s, a, s') \right. \\ & \left. + p^{(3)}(s, a, s') D^{(3)}(s, a, s') \right] J^\alpha(s') \\ = & \sum_{s' \in S_2} \left[ p(s, a, s') D(s, a, s') + \sum_{\hat{s} \in S_1} p(s, a, \hat{s}) D(s, a, \hat{s}) \{ p(\hat{s}, b, s') D(\hat{s}, b, s') \right. \\ & \left. + \sum_{\hat{s}' \in S_1} p(\hat{s}, b, \hat{s}') D(\hat{s}, b, \hat{s}') p(\hat{s}', b, s') D(\hat{s}', b, s') \} \right] J^\alpha(s') \\ = & \sum_{s' \in S_2} p(s, a, s') D(s, a, s') J^\alpha(s') + \sum_{\hat{s} \in S_1} p(s, a, \hat{s}) D(s, a, \hat{s}) \end{aligned}$$

$$\begin{aligned}
& \cdot \sum_{s' \in S_2} [p(\hat{s}, b, s')D(\hat{s}, b, s')] \\
& + \sum_{\hat{s}' \in S_1} p(\hat{s}, b, \hat{s}')D(\hat{s}, b, \hat{s}')p(\hat{s}', b, s')D(\hat{s}', b, s')]J^\alpha(s') \tag{A.4}
\end{aligned}$$

The second summation in the second term of the above formula is

$$\begin{aligned}
X(\hat{s}) &= \sum_{s' \in S_2} p(\hat{s}, b, s')D(\hat{s}, b, s')J^\alpha(s') \\
&+ \sum_{\hat{s}' \in S_1} p(\hat{s}, b, \hat{s}')D(\hat{s}, b, \hat{s}') \cdot \sum_{s' \in S} p(\hat{s}', b, s')D(\hat{s}', b, s')J^\alpha(s') \tag{A.5}
\end{aligned}$$

due to the condition that  $p(\hat{s}', b, s') = 0$  for all  $s' \in S_1$ . Because  $A(\hat{s}') = \{b\}$ , and  $c(\hat{s}', b) = 0$ ,

$$\begin{aligned}
J^\alpha(\hat{s}') &= \max_{b \in A(\hat{s}')} \left\{ c(\hat{s}', b) + \sum_{s' \in S} p(\hat{s}', b, s')D(\hat{s}', b, s')J^\alpha(s') \right\} \\
&= \sum_{s' \in S} p(\hat{s}', b, s')D(\hat{s}', b, s')J^\alpha(s') \tag{A.6}
\end{aligned}$$

Combining (A.5), (A.6),  $A(\hat{s}) = \{b\}$ , and  $c(\hat{s}, b) = 0$ :

$$\begin{aligned}
J^\alpha(\hat{s}) &= \max_{b \in A(\hat{s})} \left\{ c(\hat{s}, b) + \sum_{s' \in S} p(\hat{s}, b, s')D(\hat{s}, b, s')J^\alpha(s') \right\} \\
&= \sum_{s' \in S} p(\hat{s}, b, s')D(\hat{s}, b, s')J^\alpha(s') \\
&= X(\hat{s}) \tag{A.7}
\end{aligned}$$

From (A.3), (A.4), and (A.7), we have for all  $s \in S_2$

$$\pi^\alpha(s) = \arg \max_{a \in A(s)} \left\{ c(s, a) + \sum_{s' \in S} \left[ p(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{s,s'}(\xi|a) \right] J^\alpha(s') \right\} \tag{A.8}$$

(A.7) together with (A.8) and by the uniqueness of the optimal value function, it is easy to verify that  $J^\alpha(s) = J_0^\alpha(s), \forall s \in S$ . Therefore, by (A.1), (A.2), and (A.8)

$$\pi^\alpha(s) = \pi_0^\alpha(s), \quad \forall s \in S \tag{A.9}$$

In the above proof, e.g. (A.4), we used the memoryless property of the transition processes. ■

In CAC, states such as  $s_0 = (\mathbf{x} = \mathbf{e}_i, -\mathbf{e}_i)$  can serve as  $\hat{s}'$ . Since from  $s_0$ , it is not possible to have any more call departures. And due to the capacity constraint, from any state  $s$ , under any action  $a$ , it will take at most finite number of *consecutive* call

departures to reach a state like  $s_0$ . Theorem 2 can be generalized in the following way.

COROLLARY 1. *If  $m_{sa}$  is a random number, such that*

$$P\{0 < m_{sa} < \infty\} = \sum_{m_{sa}=1}^{\infty} p(m_{sa}) = 1,$$

*then Theorem 2 still holds.*

**Proof:** If  $m_{sa}$  is random, then (A.3) can be re-written as

$$\begin{aligned} \pi^\alpha(s) = \arg \max_{a \in A(s)} & \left\{ c(s, a) + \sum_{s' \in S_2} \sum_{m_{sa}=1}^{\infty} p(m_{sa}) \right. \\ & \cdot \left[ p(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a, N=1) \right. \\ & + p^{(2)}(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a, N=2) + \dots \\ & \left. \left. + p^{(m_{sa}+1)}(s, a, s') \int_0^\infty e^{-\alpha\xi} dF_{ss'}(\xi|a, N=m_{sa}+1) \right] J^\alpha(s') \right\} \end{aligned}$$

Using similar steps as in the proof of Theorem 2 for each finite  $m_{sa}$ , we obtain the desired result. ■

## References

- Altman, E., & Schwartz, A. (1991). Adaptive control of constrained Markov chains. *IEEE Trans. on Auto. Control*, 36(4), 454–462.
- Bertsekas, D.P., & Gallager, R. (1992). *Data Networks*, 2nd Ed., Prentice Hall.
- Bertsekas, D.P., & Tsitsiklis, J.N. (1996). *Neural-Dynamic Programming*, Athena Scientific.
- Beutler, F.J., & Ross, K.W. (1985). Optimal policies for controlled Markov chains with a constraint. *J. Math. Anal. Appl.*, 112, 236–252.
- Beutler, F.J., & Ross, K.W. (1986). Time-average optimal constrained semi-Markov decision processes. *Adv. Appl. Prob.*, 18, 341–359.
- Boyan, J.A., & Littman, M.L. (1994). Packet routing in dynamically changing networks: a reinforcement learning approach. Cowan, J.D., et al., ed. *Advances in NIPS 6*, Morgan Kaufman, SF, (pp. 671–678).
- Brown, T.X (1997). Adaptive statistical multiplexing for broadband communications. Invited Tutorial *Fifth IFIP Workshop on Performance Modeling & Evaluation of ATM Networks*, Ilkley, U.K. Published in ed. Kouvatsos, D., *Performance Evaluation and Application of ATM Networks*, Kluwer, 2000, pp. 51–79.
- Brown, T.X (2001). Statistical classification based admission control. *Proc. SPIE Conf. on Internet Performance and Control of Network Systems II*, ed. R. D. van der Mei et al., 4523, 1–14.
- Brown, T.X, Tong, H., & Singh, S. (1999). Optimizing admission control while ensuring quality of service in multimedia networks via reinforcement learning. *Advances in Neural Information Processing Systems 11*, ed. M. Kearns et al., MIT Press (pp. 982–988).

- Carlstrom, J., & Nordstrom, E. (1997). Control of self-similar ATM call traffic by reinforcement learning. *Applications of Neural Networks to Telecommunications 3*, ed. J. Alspector et al., LEA Publishers.
- Dziong, Z., & Mason, L.G. (1994). Call admission and routing in multi-service loss networks. *IEEE Trans. Commun.*, 42(4), 2011–2022.
- Dziong, Z. (1997). *ATM Network Resource Management*, McGraw-Hill.
- Feinberg, E.A. (1994). Constrained semi-Markov decision processes with average reward. *ZOR-Math. Methods Oper. Res.*, 39, 257–288.
- Gabor, Z., Kalmar, Z., & Szepesvari, C. (1998). Multi-criteria reinforcement learning. *Proc. International Conference on Machine Learning*, Madison, WI.
- Krishnan, K.R. (1990). Markov decision algorithms for dynamic routing. *IEEE Commun. Mag.*, pp. 66–69, Oct.
- Lee, W.C., Hluchyj, M.G., Humblet, P.A. (1995). Routing subject to quality of service constraints in integrated communication networks. *IEEE Network*, pp. 46–55, July/August.
- Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.V. (1993). On the self-similar nature of Ethernet traffic. *Proc. of ACM SIGCOMM*, pp. 183–193.
- Marbach, P., & Tsitsiklis, J.N. (1997). A Neuro-Dynamic Approach to Admission Control in ATM Networks: The Single Link Case. *ICASSP*.
- Marbach, P., Mihatsch, O., & Tsitsiklis, J.N. (2000). Call admission control and routing in integrated services networks using neuro-dynamic programming. *IEEE J. Sel. Areas Commun.*, 18(2) 197–208
- Mitra, D., Reiman, M. I., & Wang, J. (1998). Robust dynamic admission control for unified cell and call QoS in statistical multiplexers. *IEEE J. Sel. Areas Commun.*, 16(5), 692–707.
- Nie, J., & Haykin, S. (1999). A Q-learning based dynamic channel assignment technique for mobile communication systems. To appear in *IEEE Trans. on Vehicular Technology*, 48(5), 1676–1687.
- Pornavalai, C., Chakraborty, G., Shiratori, N. (1997). Qos based routing algorithm in integrated services packet networks. *Proc. 5th Intl. Conf. on Network Protocols*, Atlanta, GA, pp. 167–174.
- Singh, S.P., & Bertsekas, D.P. (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. *Advances in NIPS 9*, ed. Mozer, M., et al., MIT Press, (pp. 974–980).
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning, An Introduction*, The MIT Press.
- Szepesvari, Cs. (1998). The asymptotic convergence-rate of Q-learning. Jordan, M., et al., ed. *Advances in NIPS 10*, MIT Press (pp. 1064–1070).
- Tong, H. (1999). *Adaptive Admission Control and Routing under Quality of Service Constraints in Broadband Communications*, Ph.D. thesis, Univ. of Colorado, Dept. of Electrical and Computer Engineering, Boulder, CO.
- Tong, H., & Brown, T.X (1998). Estimating loss rates in an integrated services network by neural networks. *Proc. IEEE Globecom98*, Sydney, Australia.
- Tong, H., & Brown, T.X (2000). Adaptive call admission control under quality of service constraints: a reinforcement learning solution. *IEEE J. Sel. Areas Commun.*, 18(2), 209–221.
- Watkins, C.J.C.H., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8, 279–292.