Reinforcement Learning for Control of Self-Similar Call Traffic in Broadband Networks

Jakob Carlström and Ernst Nordström

Department of Computer Systems, Uppsala University, Uppsala, Sweden
Email: jakobc@DoCS.UU.SE, ernstn@DoCS.UU.SE

Reinforcement learning is applied to admission control of self-similar call traffic in broadband networks. The reinforcement learning method solves a Markov Decision Problem without the need for a model of the dynamics of the controlled system. A state descriptor containing continuous-valued running averages of the call inter-arrival times is employed. Radial-basis function neural networks approximate the value function. In simulations, the proposed method yields higher throughput than methods that do not exploit the self-similarity of the call arrival process.

1. INTRODUCTION

In broadband communications networks for integrated services, such as Asynchronous Transfer Mode (ATM) networks, resource control is of crucial importance for the network operator as well as the network users. The objective of resource control is to maintain the service quality (Grade of Service) for network calls while maximizing the network operator's revenue. At the call level, service quality is measured in terms of call blocking probabilities, and the resource to control is bandwidth. If a uniform call charging policy is used, the revenue is proportional to the long-term bandwidth utilization.

Reinforcement learning [1, 2] offers a way of finding optimal solutions to Markov Decision Problems by iterative refinement of the control policy. Reinforcement learning algorithms sample the immediate rewards (payoffs) upon state transitions in the controlled system or in a simulation model. Unlike dynamic programming [3], no model of system state transition probabilities and expected immediate rewards is needed. Thus, reinforcement learning is suitable where such a model is expensive to obtain. Further, by combining reinforcement learning with parametric function approximators (such as neural networks), Markov Decision Problems with very large or even infinite state spaces may be approximately solved by reinforcement learning.

In [4, 5], we proposed a method based on reinforcement learning for link allocation, a subproblem of network routing. In simulations with Poisson call arrival processes, this method outperformed standard heuristics, and achieved performance similar to that of Dziong's and Mason's dynamic programming solution [6]. Marbach et al. have shown how to extend the use of reinforcement learning to network routing [7], also assuming Poisson call arrival processes.

In this paper, we do not assume Poisson call arrival processes. Measurements of network arrivals have displayed many examples of statistically *self-similar* arrival processes [8, 9, 10]. This traffic is bursty on several time-scales, a feature that should be considered in the design of effi-

cient resource control methods. We present a link admission controller for such arrival processes. The controller is based on reinforcement learning with neural networks, employing a state descriptor containing the number of active calls per service class along with measures of the history of the arrival processes.

Although this paper concentrates on the link admission control problem, the link admission controller we describe may be used as a building block for optimal link allocation or routing, as shown in [4, 6, 7].

2. SELF-SIMILAR CALL ARRIVAL PROCESSES

The limitations of the traditional Poisson model for network arrival processes have been demonstrated in a number of studies [9, 10]. Poor correspondence with the Poisson process has been shown for a range of traffic types that often are modelled as being Poisson, for example arrivals of HTTP requests and machine-initiated TCP connections. This difference is manifested by heavy-tailed inter-arrival time distributions and long-term correlations in the arrival processes. Various self-similar (fractal-like) models for such processes have been shown to correspond better with this traffic.

In this paper, control of self-similar call arrival processes in a broadband multi-rate networks is studied. One example of where this situation occurs is an ATM network carrying TCP/IP traffic with self-similar connection arrivals.

2.1. Properties of Self-Similar Traffic

A self-similar arrival processes has no natural burst length. On the contrary, its arrival intensity varies considerably over many time scales. This makes the variance of its sample mean decay slowly with the sample size, and its auto-correlation function to decay slowly with time, compared to Poisson traffic [10].

The complexity of control and prediction of Poisson traffic is reduced by the memory-less property of the Poisson process: its expected future depends entirely on the (constant) arrival intensity, not on the history of the process. On the other hand, the long-range dependence of self-similar traffic makes it possible to improve the predictions of the process future by observing the process history. Norros [11] gives a rule-of-thumb for predicting one type of self-similar processes (fractional Brownian motion): “one should predict (with the appropriate nonuniform weights) the next second with the latest second, the next minute with the latest minute, etc.”

A compact statistical measure of the degree of self-similarity of a stochastic process is the *Hurst parameter* [8]. For self-similar traffic this parameter takes values in the interval (0.5, 1], whereas Poisson processes have a Hurst parameter of 0.5.

2.2. Traffic Synthesis by The Fractional ARIMA Process

There exist many methods for synthesis of self-similar processes. For an overview, see for instance [8, 10]. In this paper, synthetic traffic traces are generated from a Gaussian fractional ARIMA (0, d, 0) model [8].

First, N_0 zero-expectance continuous values from a fractional ARIMA (auto-regressive integrated moving average) process Z_i are generated by a convolution formula:

$$Z_i = \sum_{j=0}^{N_0} c_{i-j} e_j, \quad i \in \{0, ..., N_0\}, \tag{1}$$

where e_j are the normally distributed innovations of the process, having mean 0 and variance 1. The coefficients c_j are given by:

$$c_j = \frac{\Gamma(j + d)}{\Gamma(d)\Gamma(j + 1)}, \quad j \in \{0, ..., N_0\}, \tag{2}$$

where Γ is the standard gamma function. The c_j's can be computed recursively:

$$c_{j+1} = \frac{j + d}{j + 1} c_j, \qquad c_0 = 1, \quad j \in \{0, ..., N_0\}. \tag{3}$$

The resulting process is asymptotically self-similar, with Hurst parameter $H = d + 0.5$, $0 < d < 0.5$. The increments $\tilde{Z}_i$ are then generated from Z_i by adding a constant d, truncating negative values to zero, and rounding off to integers. Finally, the duration of the basic time slot is fixed, and $\tilde{Z}_i$, $i = 1, ..., N_0$, individual arrival times are drawn from a uniform distribution within their respective time slots. Thus, the resulting traffic trace contains $\Sigma_{i=1}^{N_0} \tilde{Z}_i$ arrival times. The choice of the duration of the time slot determines the intensity of the arrival process.

3. THE LINK ADMISSION CONTROL PROBLEM

In the link admission control problem, a link with capacity C [units/s] is offered calls from K different service classes. Calls belonging to such a class $j \in J = \{1, ..., K\}$ have the same bandwidth requirements b_j [units/s]. In the set-up procedure, the terminal that requests the new call declares what service class the new call belongs to.

We focus on arrival processes with memory, so we assume that calls arrive according to self-similar processes, and that their holding times are exponentially distributed with mean $1/\mu_j$ [s].

Access to the link is controlled by a *policy* π that maps *states* $x \in X$ to *actions* $a \in A$, $\pi: X \rightarrow A$. The set X contains all feasible link states, and the action set A is

$$A = \{(a_1, a_2) : a_j \in \{0, 1\}, j \in J\},$$

where a_j is 0 for rejecting a new class-j call and 1 for accepting it. The set of link states is given by $X = N \times H$, where N is the set of feasible call number tuples, and H is the Cartesian product of some representations, h_j, of the histories of the per-class call arrival processes (for the memory of self-similar arrival processes). N is given by

$$N = \left\{ n = (n_1, ..., n_K) : n_j \geq 0, \ j \in J; \ \sum_{j \in J} n_j b_j \leq C \right\},$$

where n_j is the number of type-j calls accepted on the link.

The objective of the link admission controller is to learn a policy that maximizes the long-term reward (revenue) on the link. By taking optimal actions, the policy controls the (unknown) probabilities of state transitions so as to increase the probability of reaching states that yield high long-term rewards. We assume a uniform call charging policy, which means that the reward rate $\varrho(t)$ at time t is equal to the carried bandwidth at time t:

$$\varrho(t) = \sum_{j \in J} n_j(t) b_j \tag{4}$$

Time evolves continuously, with discrete arrival and departure events $k = 0, 1, 2...$ The discounted immediate reward, $r(x_k)$, received between entering a state x_k at time t_k and the next state x_{k+1} at time t_{k+1}, is

$$r(x_k) = \int_{t=t_k}^{t_{k+1}} e^{-\beta(t-t_k)} \varrho(t) dt = \frac{1}{\beta}\left[1 - e^{-\beta(t_{k+1}-t_k)}\right] \varrho(t_k) , \tag{5}$$

where $\varrho(t_k)$ is the reward rate after taking an action at time t_k. The exponential discounting of rewards, using a constant $\beta > 0$, prevents the sum of future expected rewards (assuming $x_0 = x$ and $t_0 = 0$) from growing towards infinity. The *value function* $V_\pi(x)$ estimates the long-term reward under a policy π:

$$V_\pi(x) = \lim_{N \to \infty} E\left\{ \sum_{k=0}^{N} e^{-\beta t_k} r(x_k) \;\middle|\; x_k = x \right\}, \quad x \in X \tag{6}$$

The network availability constraint (limited call blocking probabilities) is currently not considered in the objective function.

A feature of optimal link admission control is “intelligent blocking”. This typically occurs when a link has a free capacity equal to the bandwidth of a wide-band call. By rejecting a narrow-band call request, the controller reserves bandwidth for the next wide-band call, in order to increase the expected long-term reward.

4. REINFORCEMENT LEARNING SOLUTION

This section describes our reinforcement learning method for link admission control. Before presenting the method, we show how to incorporate statistics from the arrival process history into the link state descriptor.

4.1. Sufficient Statistics from The Process History

Statistics from previous observations of the arrival process for calls from a class j are represented by *history vectors* $h_j \in H_j$. As discussed by Bertsekas [3], these should fulfill the following two requirements:

1. Preservation of the information that is essential for control
2. A constant, small vector dimension

The first requirement is known as *sufficient statistics*. The second one facilitates learning by limiting the size of the state space to be explored. Along with the call-number tuples, the sufficient statistics should constitute approximations of Markovian states, i.e., no information which is useful for predicting the future of the system is discarded.

We chose the following ansatz to representing the history of the arrival process: for all classes $j \in J$, M running averages $h_j = (h_{j1}, ..., h_{jM})$ of the inter-arrival times were computed recursively using forgetting factors $\alpha_1, ..., \alpha_M$:

$$h_{ji}(k) = \alpha_i \Big[t_j(k) - t_j(k-1) \Big] + (1 - \alpha_i) h_{ji}(k-1) , \tag{7}$$

where $t_j(k)$ is the arrival time of the k-th call from class j. This choice was supported by experiments with prediction of short-term arrival rates. We trained a feed-forward neural network on synthetic traffic traces with h as input. The network learnt to make good predictions of the future short-term arrival rate. This shows that (n, h) serve as approximate sufficient statistics. Another argument for using mean inter-arrival times in the state descriptor is the strong dependence of the optimal policy on the arrival rates in control of constant-rate traffic, e.g. Poisson traffic.

By matching the forgetting factors α_i to the distribution of the inter-arrival times, the computation of the inter-arrival time estimates can be made to comply with Norros' rule of thumb, quoted in section 2.1 This is done by setting

$$1 - \alpha_i \approx e^{-\beta/\bar{\lambda}} , \tag{8}$$

where $1/\bar{\lambda}$ is the mean per-class inter-arrival time in the system. This means that the discounting window used for future rewards, and thereby inter-arrival times, is the same as the discounting window used for historical inter-arrival times. Heuristically, some α_i are set to values larger than $1 - e^{-\beta/\bar{\lambda}}$, and some α_i to smaller values.

4.2. Temporal-Difference Learning with Neural Networks

When the link admission controller makes a decision in a state x, it is always possible to determine what new configuration x' will occur *immediately after* the action is performed. For example, if a class-j call arrives and the current state is $x = (n_1, ..., n_j, ..., n_K, h)$, the state will be unchanged after a reject action, whereas it will be $x' = (n_1, ..., n_j + 1, ..., n_K, h)$ after an accept action (provided $x' \in X$). By learning values of such afterstates [1] (henceforth denoted with a prime $'$), instead of state-action pairs (x, a) as in e.g. Q-learning [1], the value function can be defined over a smaller input domain, provided that the mapping from state-action pairs to afterstates is many-to-one. This is the case in the link admission control problem.

Due to the continuous-valued history vectors, the number of afterstates is infinite, making lookup-table storage of the value function impossible. Instead, a parametric value function $V(x')$, that models $V_\pi(x')$ in equation 6, is realized by feed-forward neural networks with normalized Gaussian radial-basis function (RBF) units [13]. For each $n \in N$, a RBF network with MK inputs approximates the value function V_n from the history vector $h = (h_{11}, ..., h_{MK})$:

$$V_n(h) = \sum_{i=1}^{S} w_i \, f_i(h) , \tag{9}$$

where S is the number of basis functions, w_i are the network's weights, and f_i are the basis functions:

$$f_i(h) = \exp\left(-\frac{\|h - v_i\|^2}{\sigma_i^2}\right). \quad (10)$$

The *center* vectors v_i are MK-dimensional and the *width* parameters σ_i are scalar.

To simplify computations, the radial-basis functions are factorized. This is done by computing the activation of a basis function with MK inputs as the product of K basis functions (one per service class) with M inputs each. Further, the outputs of the RBF's are normalized class-wise to the sum of 1 before multiplication (not shown in equation (10)). The center and width parameters of the RBFs are chosen after inspection of the data set to cover the history vector space.

The link admission controller learns values by Sutton's TD(0) temporal-difference learning rule [1]. The policy $\pi(x)$ is implicitly defined by the value function. During learning, actions are selected stochastically, with high probability for the action resulting in the afterstate with the highest associated values, using epsilon-greedy action selection [12]. As learning proceeds, the degree of randomness is successively decreased, until finally the policy deterministically selects the action that results in the highest after-state value. This trial-and-error search allows the controller to explore the state-action space and improve the policy.

After every decision, the value $V(x'_k)$ of the previous after-state is adjusted by updating the weights of the RBF network on the temporal-difference error, $\varepsilon(k)$:

$$\varepsilon(k) = r(x'_k) + e^{-\beta(t_{k+1}-t_k)}V(x'_{k+1}) - V(x'_k) \quad (11)$$

using the same definitions as in section 3, but for afterstates instead of states. The least mean squares (LMS) update rule gives, for the weights of the RBF network that corresponds to the call number tuple n_k (a part of x_k):

$$w_i(k+1) = w_i(k) + \eta\varepsilon(k)\phi_i(h_k), \quad i \in \{1, ..., S\}, \quad (12)$$

where h_k is the arrival history descriptor included in x_k and η is a step size parameter.

The value function is updated on transitions caused by call arrivals as well as call departures. For obvious service reasons, call departures must always be accepted.

5. SIMULATION RESULTS

The performance of the reinforcement learning method was evaluated in simulations, where the link admission controller adapted to synthetic call traffic.

5.1. Synthetic Traffic Traces

We generated synthetic traffic traces containing arrival/departure pairs from two call classes ($K = 2$), characterized by bandwidth requirements $b_1 = 1$ (narrow-band) and $b_2 = 6$ (wide-band) [units/s] and call holding times with mean $1/\mu_1 = 1/\mu_2 = 1$ [s].

The self-similar call arrival processes were generated from a fractional ARIMA process with Hurst parameter 0.85. The offset δ was set to 4.0, and the inter-arrival times were linearly scaled so that the mean long-term arrival rates $\bar{\lambda}_1$ and $\bar{\lambda}_2$ for the two classes fulfil

$$\frac{b_1\bar{\lambda}_1}{\mu_1} + \frac{b_2\bar{\lambda}_2}{\mu_2} = \gamma C \tag{13}$$

where C is the capacity of the communication link and γ is the mean relative load. Traces with $\gamma = 1.0$, 1.2 and 1.5 were generated. The arrival rate ratio $\bar{\lambda}_1/\bar{\lambda}_2$ was varied from 0.4 to 2.0.

5.2. Link Admission Control

We evaluated the performance of our proposed link admission controller in simulations using the OPNET Modeler simulation software [14]. The simulated communication link had a capacity of C = 24 [units/s], and was offered calls from the self-similar arrival processes.

For comparison, we repeated the simulations with two other link admission controllers. The first one was a tabular reinforcement learning controller, using a lookup table instead of a neural network for the value function. The second one used complete sharing: to accept a call if the free capacity on the link is sufficient, i.e. no intelligent blocking.

The tabular reinforcement learning controller assumes Poisson arrival processes. From this assumption, it follows that the call number tuples $n \in N$ constitute Markovian states. Consequently, the value function table stores only one value per n.

The discount constant β of both the reinforcement learning controllers was set to 2.0. This value was chosen because it yields an approximate discounting of the value of the next state in the TD(0) update rule (equation 11) of 0.9 — a typical value of the discount parameter in discrete-time reinforcement learning. It also worked well in simulations. The step size η was linearly lowered from 0.09 to 0.01 during training.

For the reinforcement learning controller employing neural networks, two-dimensional ($M = 2$) history vectors h_j were computed. The per-class arrival rate in the simulations varied from 1.5 to 9.0. From this and equation (8), forgetting factors (α_1, α_2) = (0.1, 0.4) were determined. This choice was supported by simulations. The RBF networks (one per $n \in N$) had 64

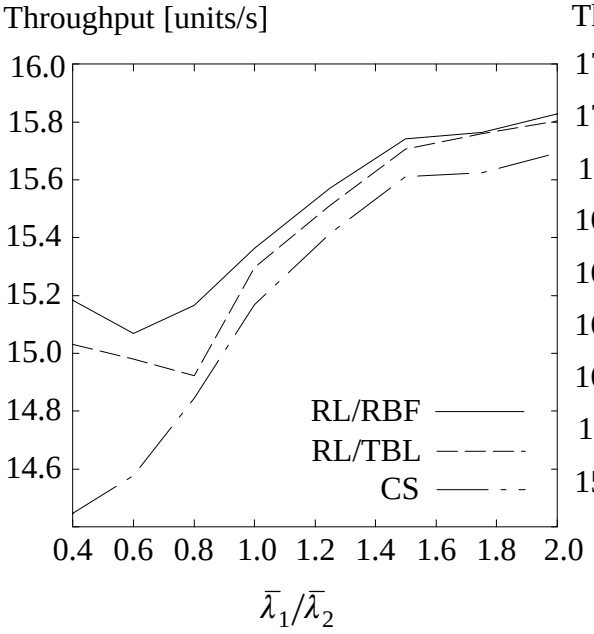

Figure 1. Throughput versus arrival rate ratio for γ = 1.0.

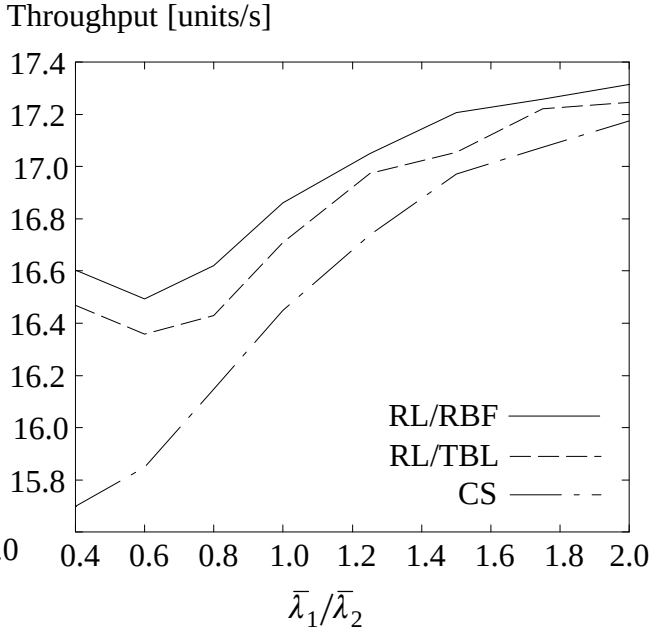

Figure 2. Throughput versus arrival rate ratio for γ = 1.25.

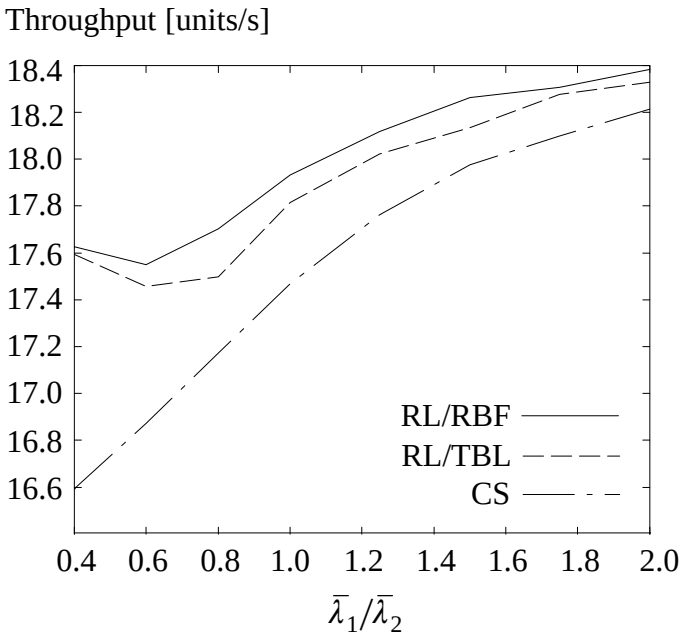

Figure 3. Throughput versus arrival rate ratio for γ = 1.5.

radial-basis functions, factorized to 8 units per call class, plus a bias activation. Its weights were initialized to favor acceptance of all calls in all states.

The reinforcement learning controllers were allowed to adapt to the first 400 000 simulated arrivals of the traffic traces. The performance of all three methods was measured on the subsequent 400 000 arrivals.

Figure 1, 2 and 3 show throughput versus arrival rate ratio for $\gamma = 1.0$, 1.25 and 1.5, respectively. Each data point is the averaged throughput for 10 traffic traces. Reinforcement learning with RBF neural networks (RL/RBF) yields up to 1.6% better performance than tabular reinforcement learning (RL/TBL). Complete sharing (CS) consistently yields the lowest throughput; up to 6.2% worse than RL/RBF.

The difference in throughput between reinforcement learning and complete sharing is highest for low arrival rate ratios. The reason for this is that the throughput increase by reserving bandwidth for high-rate calls is considerably higher than the loss of throughput from the blocked low-rate narrowband traffic.

Figure 4 shows how the values learnt by the RBF network for $\gamma = 1.5$, $(n_1, n_2) = (6, 2)$, vary with the long- and short-term history. The history parameters of the narrow-band class is kept constant at typical values, $h_{11} = h_{12} = 0.2$, whereas those of the wide-band class are varied. The values are high when the mean of previous inter-arrival times for wide-band calls is small, probably because of correlations with future short inter-arrival times and thereby large potential rewards. These variations are not possible with the tabular reinforcement learning method, which stores only one value per combination of per-class calls (n_1, n_2).

Figure 5 shows the resulting border (where the difference $V_{(7,\,2)} - V_{(6,\,2)}$ is zero) between intelligent blocking and not blocking of narrowband calls in (6, 2) for three different arrival rate ratios and $\gamma = 1.5$. In the plot, the long- and short-term history parameters are kept equal to project the four-dimensional history vector space onto two dimensions. The curves are labeled

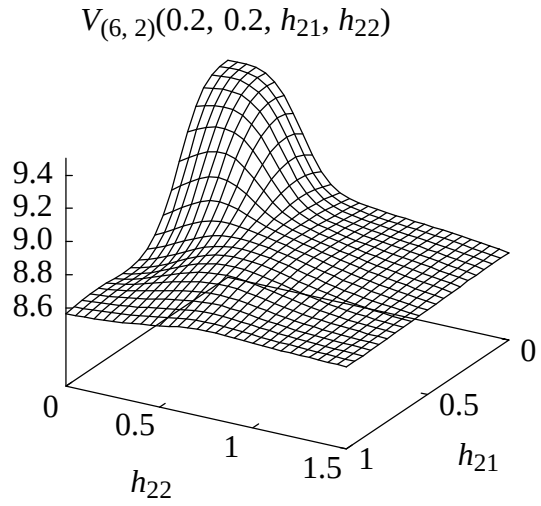

Figure 4. The value function for varying history of the wide-band arrivals process (h_{21} = long-term history, h_{22} = short-term history) for a fixed tuple of active calls and a fixed narrow-band arrival history.

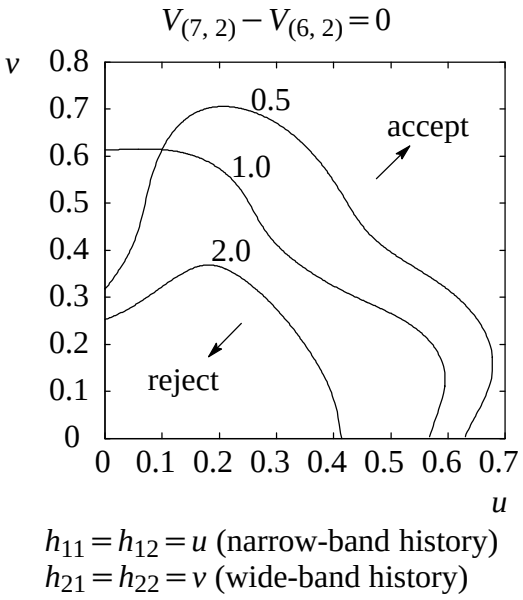

Figure 5. The border between accepting and rejecting (intelligent blocking)narrow-band calls for various ratios of arrival rates. The value by each borderline indicates the ratio between call arrival rates.

with the ratios $\bar{\lambda}_1/\bar{\lambda}_2$ between the rates of call arrival in the traffic traces. The difference is negative below and to the left of the curves (implying rejection of for narrow-band calls), and positive above and to the right (implying acceptance). The rejection region is smaller for higher ratios, where narrow-band calls are more probable.

6. CONCLUSION

We have presented a new method for link admission control of calls from self-similar arrival processes. It learns a policy by reinforcement learning, using radial-basis function neural networks for approximation of the value function. In simulations, this method increased the throughput slightly compared to a controller based on tabular reinforcement learning, and considerably compared to a controller using complete sharing. These throughput differences are likely to increase if the controllers are used as building blocks for link allocation or routing, where the values, not just the policy, affect decisions.

This work has demonstrated the capability of reinforcement learning to make use of parameters that are relevant for a control problem, without explicitly modelling their influence on the dynamics of the controlled system. It has also shown some limitations of modelling self-similar traffic by Poisson processes in network control.

The issue of learning time and adaptation to varying arrival rate ratios was not particularly addressed here. However, the reinforcement learning method with neural networks needs rela-

tively long time to converge. It therefore seems to be better suited for off-line learning on recorded traffic traces than to on-line learning. If the characteristics of the arrival process do not change too frequently, on-line refinement of the policy may still be possible.

ACKNOWLEDGEMENTS

The authors wish to thank Olle Gällmo and Lars Asplund for valuable comments on the manuscript.

REFERENCES

1. R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.
2. D.P. Bertsekas and J.N. Tsisiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, Mass., 1996.
3. D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, Athena Scientific, Belmont, MA, 1995.
4. E. Nordström and J. Carlström, "A Reinforcement Learning Scheme for Adaptive Link Allocation in ATM Networks", in *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 2* (IWANNT*95), Alspector, Goodman, Brown (eds.), Lawrence Erlbaum, pp. 88–95, 1995.
5. E. Nordström, J. Carlström. O. Gällmo, L. Asplund, "Neural Networks for Adaptive Traffic Control in ATM Networks", *IEEE Communications Magazine*, vol. 33 no. 10, pp. 43–49, Oct. 1995.
6. Z. Dziong and L. Mason,"Call Admission Control and Routing in Multi-service Loss Networks", *IEEE Transactions on Communications*, vol. 42, no. 2, pp. 2011–2022, Feb. 1994.
7. P. Marbach, O. Mihatsch, M. Schulte and J.N. Tsitsiklis, "Reinforcement Learning for Call Admission Control and Routing in Integrated Service Networks", in *Advances in Neural Information Processing Systems 9*, MIT Press, due May 1998.
8. J. Beran, *Statistics for Long-Memory Processes*, Monographs on Statistics and Applied Probability 61, Chapman & Hall, 1994.
9. V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", in Proc. of SIGCOMM 94 –8/94, London, UK, pp. 257–268, 1994.
10. W.E. Leland, M.S. Taqqu, W. Willinger and D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.
11. I. Norros, "On the Use of Fractional Brownian Motion in the Theory of Connectionless Networks", *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 6, pp. 953–962, Aug 1995.
12. J. Carlström and E. Nordström, "Control of Self-Similar ATM Call Traffic by Reinforcement Learning", in *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 3* (IWANNT*97), Alspector, Goodman, Brown, (eds.), Lawrence Erlbaum, pp. 54–62, 1997.
13. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Co., Englewood Cliffs, NJ, 1994.
14. *OPNET Modeler documentation*, MIL3, Inc., Washington, D.C., 1998.