A NEURO-DYNAMIC PROGRAMMING APPROACH TO ADMISSION CONTROL IN ATM NETWORKS: THE SINGLE LINK CASE

Peter Marbach and John N. Tsitsiklis

Laboratory for Information and Decision Systems, MIT
Cambridge, MA 02139, USA

ABSTRACT

We are interested in solving large-scale Markov Decision Problems. The classical method of Dynamic Programming provides a mathematical framework for finding optimal solutions for a given Markov Decision Problem. However, for Dynamic Programming algorithms become computationally infeasible when the underlying Markov Decision Problem evolves over a large state space. In recent years, a new methodology, called Neuro-Dynamic Programming, has emerged which tries to overcome this "curse of dimensionality". We present how Neuro-Dynamic Programming can be applied to the Admission Control Problem for a single link in an ATM environment. Based on results obtained through Neuro-Dynamic Programming, we derive a heuristic "Threshold" policy. Performances of the policies obtained through Neuro-Dynamic Programming are compared with a policy which always accepts a customer when the required resources are available.

1. INTRODUCTION

Markov Decision Problems have been a popular paradigm for sequential decision making under uncertainty. Dynamic Programming [1] provides a framework for studying such problems, as well as algorithms for computing optimal decision policies. Unfortunately, these algorithm become computationally infeasible for Markov Decision Problems which evolve over a large state space. This so called "curse of dimensionality" renders the classical methods of Dynamic Programming inapplicable to most realistic problems.

In recent years, the new methodology Neuro-Dynamic Programming [2] has emerged. Neuro-Dynamic Programming tries to overcome the curse of dimensionality by employing stochastic approximation algorithms and functional approximation techniques such as neural networks. The outcome is a methodology for approximating Dynamic Programming solutions in a computationally feasible manner.

Over the past few years, methods of Neuro-Dynamic Programming have been applied successfully to challenging problems. Examples include a program that implements a Backgammon player [3], an elevator dispatcher [4], a job scheduling problem [5] and control policy for admission control in wireless communication networks [6]. Although methods of Neuro-Dynamic Programming have been successfully applied to challenging problem, most algorithms proposed in the field are not well understood at a theoretical level. Nevertheless, the potential of these methods for solving systematically large-scale Markov Decision Problems and the successful experimental work in the field has drawn considerable attention.

In this paper, we illustrate the ideas and methods of Neuro-Dynamic Programming using the example of Admission Control in an ATM network. In particular, we consider a single communication link with a given bandwidth.

The paper is structured as follows: after stating the admission control problem precisely in section 2, we will in section 3 solve it formally using the classical method of Dynamic Programming. Section 4 introduces the TD(0) algorithm of Neuro-Dynamic Programming. In section 5, we present experimental results.

2. ADMISSION CONTROL

In this section, we formulate more precisely the Admission Control Problem for a single link.

We are given a single communication link with a total bandwidth of B units. We intend to support a finite set of customer classes $\{1, 2, .., N\}$. Customers of the different classes request connections over the communication link according to independent Poisson Processes. The Poisson Process associated with customer class n is charaterized by the arrival rate parameter $\lambda(n)$. Once accepted, a customer of class n seizes $b(n)$ units of bandwidth for t units of time, where t is exponentially distributed with parameter $\nu(n)$, independently of everything else in the system.

One possible way of executing admission control would be to always accept a customer if the required bandwidth $b(n)$ is available, and otherwise to reject it. A more sophisticated way is to reject a customer even when there is enough free bandwidth available. One may do this in order to reserve some bandwidth for customers of a class which is considered to be especially important. To capture this idea, we associate a reward $c(n)$ with a customer of class n, i.e. whenever we accept a customer of the class n, we receive a reward of $c(n)$ units. Our objective is to maximize the discounted long term reward.

3. MARKOV DECISION PROBLEMS AND DYNAMIC PROGRAMMING

In this section, we formulate the Admission Control Problem for a single link as an infinite horizon, discounted Markov Decision Problem [1].

This research was supported by a contract with Siemens AG, Germany.

We describe the state of the communication link by an N-tuple $s = (s(1), ..., s(N))$, where $s(n)$ denotes the number of customers of class n currently using the link. The set of all possible states, i.e. the state space S, is given as follows:

$$S = \left\{ s \in R^N \;\middle|\; \sum_{n=1}^{N} s(n)b(n) \leq B, \quad s(n) \in \{0, 1, 2, ...\} \right\}$$

where $b(n)$ is the bandwidth demand of a customer of class n and B is the total available bandwidth of the communication link. Let s_t denote the state of the system at time $t \in [0, +\infty)$.

A control action $u = (u(1), ..., u(N))$ is a N-tuple such that $u(n)$ equals either 0 or 1. Given a control action u, we accept a new connection request of a customer of class n, if $u(n)$ equals 1 and reject a new customer of class n otherwise. Let U denote the set of all possible control actions:

$$U = \left\{ u \;\middle|\; u \in \{0, 1\}^N \right\}$$

We say that an event occurred in the system at time t if at time t a customer departs from the system or a new customer requires a connection over the communication link. Let t_k be the time when the kth event takes place. By convention, we start the system at time $t_0 = 0$; t_1 is the time when the first event occurs. We indentify an event by the N-tuple $\omega = (\omega(1), ..., \omega(N))$ where $\omega(n)$ equals 1 if a new customer of class n requests a connection; $\omega(n)$ equals -1 if a customer of class n departs from the system and $\omega(n)$ equals 0 otherwise. Let Ω denote the sets of all possible events:

$$\Omega = \left\{ \omega \;\middle|\; \omega \in \{-1, 0, 1\}^N, \; \sum_{n=1}^{N} |\omega(n)| = 1 \right\}$$

Let s_k be the state of the system in the interval $[t_k, t_{k+1})$. Note that if the system is in the state s_k, the probability that the next event will be a specific event ω is determined by the arrival rates $\lambda(n)$ and the departure rates $\nu(n)$.

Given a state $s \in S$, an event $\omega \in \Omega$ and a control action $u \in U$, the next state s' is given by a function $f : S \times \Omega \times U \rightarrow S$ such that if s' equals $f(s, \omega, u)$, then the following holds:

$$s'(n) = \begin{cases} s(n) & \text{if } \omega(n) = 0 \\ s(n) & \text{if } \omega(n) = 1 \text{ and } u(n) = 0 \\ s(n) + 1 & \text{if } \omega(n) = 1 \text{ and } u(n) = 1 \\ s(n) - 1 & \text{if } \omega(n) = -1 \text{ and } s(n) > 0 \\ s(n) & \text{if } \omega(n) = -1 \text{ and } s(n) = 0 \end{cases}$$

We can associate a one stage reward $g(s, \omega, u)$ with a state s, an event ω and a control action u which is given by the formula:

$$g(s, \omega, u) = \begin{cases} c(n) & \text{if } \omega(n) = 1 \text{ and } u(n) = 1 \\ 0 & \text{otherwise} \end{cases}$$

where $c(n)$ is the reward associated with the customer class n.

A stationary policy is a function $\mu : S \rightarrow U$ such that for every element s in S and every event ω in Ω, $f(s, \omega, \mu(s))$ is again a state in the state space S. Note that a stationary policy μ induces a Markov Process on the state space S. Let M denote the set of all possible policies.

With a stationary policy μ and a state s we associate the discounted reward-to-go $J_\mu(s)$:

$$J_\mu(s) = E\left[\sum_{k=0}^{\infty} e^{-\beta t_{k+1}} g(s_k, \omega_{k+1}, \mu(s_k)) \,|\, s_0 = s \right]$$

where the following condition is satisfied:

$$s_{k+1} = f(s_k, \omega_{k+1}, \mu(s_k))$$

and where β is a positive real number, called the discount rate.

Our goal is to maximize $J_\mu(s)$ simultaneously for all states s. Let μ^* be a stationary policy such that

$$J_{\mu^*}(s) \geq J_\mu(s) \text{ for all } s \in S \text{ and all } \mu \in M$$

then we call μ^* an optimal policy.

Discounted, continuous-time Markov Decision Problems can be transformed into discrete time, discounted Markov Decision Problems by a technique called uniformization [1]. By doing that, the discount rate β gets translated into a discount factor $\alpha \in [0, 1)$. Furthermore in our case, we also have to expand the set of all possible events to the set Ω':

$$\Omega' = \left\{ \omega \;\middle|\; \omega \in \{-1, 0, 1\}^N, \; \sum_{n=1}^{N} |\omega_n| \leq 1 \right\}$$

Note that we add to the set Ω the event $(0, 0, ..., 0)$ which corresponds to a self transition. For every stationary policy μ the following holds:

$$s = f(s, (0, 0, ..., 0), \mu(s))$$

Once we have transformed the continuous time Markov Decision Problem into a discrete time Markov Decision Problem, we can formulate the reward-to-go $J_\mu(s)$ for a stationary policy μ as:

$$J_\mu(s) = E\left[\sum_{k=0}^{\infty} \alpha^{k+1} g(s_k, \omega_{k+1}, \mu(s_k)) | s_0 = s \right]$$

We can also think of the reward-to-go for a stationary policy μ as a function $J_\mu : S \rightarrow R$. It is well known that the reward-to-go function for the optimal policy μ^* satisfies Bellman's equation[1]:

$$J_{\mu^*}(s) = \alpha E\left[g(s, \omega, \mu^*(s)) + J_{\mu^*}(s, \omega, \mu^*(s)) \right]$$

It is well known that Bellman's equation has an unique solution J^*, called the optimal reward-to-go function.

Given a function $J : S \rightarrow R$, we define the greedy policy of J to be the policy μ which has the following properties:

$$\mu(s) = \arg\max_{u \in U} \alpha E\left[g(s, \omega, u) + J(f(s, \omega, u)) \right]$$

[1]Note that in the Admission Control Problem, one stage rewards associated with state s_k are discounted. Therefore, Bellman's equation differs in our case from the usual form $J_{\mu^*}(s) = E\left[g(s, \omega, \mu^*(s)) + \alpha J_{\mu^*}(s, \omega, \mu^*(s)) \right]$

It is well known that the greedy policy of the optimal reward-to-go function J^* is an optimal policy. Note that knowledge of the optimal reward-to-go function J^* allows to derive an optimal policy μ^*.

In principle, an optimal policy can be found using the classical methods of Dynamic Programming. However this requires the computation and storage of $J^*(s)$ for every state s in the state space S. Note that the cardinality of the state space increases exponentially with the number of customer classes. Therefore, for Admission Control Problems which involve a large number of customer classes, using methods of Dynamic Programming is not feasible.

4. NEURO-DYNAMIC PROGRAMMING

Instead of computing the reward-to-go for every state $s \in S$, Neuro-Dynamic Programming uses a compact representation $\tilde{J}(\cdot, r)$ to approximate J^*. In particular, we approximate $J^*(s)$ by a suitable approximation architecture $\tilde{J}(s, r)$, where r is a vector of parameters, and approximate an optimal policy μ^* by the greedy policy of $\tilde{J}(., r)$. Typically, a functional approximation $\tilde{J}(., r)$ is obtained by using a neural network. The motivation behind Neuro-Dynamic is the hope that if the functional approximation $\tilde{J}(., r)$ is, in some sense, close to the optimal reward-to-go function J^*, then the greedy control policy induced by $\tilde{J}(., r)$ is, in some sense, close to an optimal control policy μ^*.

In the remainder of this section, we describe the algorithm TD(0) which we used to find a functional approximation of J^*. The TD(0) algorithm belongs to the class of Temporal Difference algorithms [7]. Temporal Difference algorithms, often reffered to as TD(λ) algorithms, are the most widely used algorithms in Neuro-Dynamic Programming. The TD(0) algorithm is a simulation based algorithm which updates the parameter vector r at every step of the simulation. Starting with an initial parameter vector r_0, TD(0) generates a sequence of parameter vectors r_k. At a simulation step t, the compact representation $\tilde{J}(., r_k)$ is used as an approximation of J^* and the greedy policy of $\tilde{J}(., r_k)$ is used as an approximation of μ^*.

Let $\tilde{J} : S \times R^K \rightarrow R$ be a family of compact representations such that $\nabla_r \tilde{J}(s, r)$ exists for every state $s \in S$ and every parameter vector $r \in R^K$. Choose an initial value of the parameter vector $r_0 \in R^K$ and an initial state $s_0 \in S$. We generate the sequence r_k by the following recursive procedure:

1. Assume we are given state s_k and parameter vector r_k; obtain the event ω_{k+1} by a simulation step of the system.

2. Choose control action $u_k \in U$ such that, for each n,:

$$u_k(n) = \begin{cases} 1 & \text{if } s_k + \mathbf{e}_n \in S \\ & \text{and } \tilde{J}(s_k, r_k) - \tilde{J}(s_k + \mathbf{e}_n, r_k) \leq c(n) \\ 0 & \text{otherwise} \end{cases}$$

   where $\mathbf{e}_n$ is the element in R^N such that the nth component is equal to 1 and all other components are equal to 0.

3. Update s_k and r_k by the following rule:

$$\begin{aligned} s_{k+1} &= f(s_k, \omega_{k+1}, u_k) \\ d_k &= \alpha \left(g(s_k, \omega_{k+1}, u_k) + \tilde{J}(s_{k+1}, r_k) \right) - \\ &\quad \tilde{J}(s_k, r_k) \\ r_{k+1} &= r_k + \gamma_k d_k \nabla_r (J(s_k, r_k)) \end{aligned}$$

   where γ_k is a small step size parameter and α is the discount factor.

4. Return to step 1.

Note that for the control action u_k, that we choose in step 2, the following holds:

$$u_k = \arg\max_{u \in U} \alpha E \left[g(s_k, \omega_{k+1}, u) + \tilde{J}(f(s_k, \omega_{k+1}, u), r_k) \right]$$

In the next section, we apply the TD(0) algorithm to the Admission Control Problem for a single communication link.

5. EXPERIMENTAL RESULTS

In this section, we present two case studies: one for a communication link which supports 3 different customer classes and one for a link which supports 10 different customer classes. In both cases, the control policy obtained by the TD(0) algorithm is compared with a policy that always accepts a new customer n if the required bandwidth $b(n)$ is available, and otherwise to rejects it. We refer to this policy as the "Always Accept" policy.

Based on the policies we obtained with the TD(0) algorithm, we designed a "Threshold" policy. The Threshold policy specifies for each customer class n a threshold $h(n)$ in units of bandwidth. If a customer of class n requests a new connection at time t over the communication link, we accept only if the used bandwidth $B(t)$ on the link at time t does not exceed the threshold $h(n)$. Simulating the policy obtained with the TD(0) algorithm, we could observe which customer classes get rejected and what percentage of customers of a particular customer class get rejected. This insight guided the tuning of the threshold parameters.

As the state space in the first case study is relatively small, we are able to compute the optimal control policy using methods of Dynamic Programming and to compare the optimal control policy with the control policy obtained by the TD(0) algorithm.

We use the average reward per time unit and the lost average reward per time unit due to customers rejection as performance measures to compare the different policies. Both averages are based on a single trajectory with 1,000,000 simulation steps, which starts with an empty system. The trajectory was generated using a random number generator which was initialized with the same seed for each evaluation.

5.1. Case Study with 3 Customer Classes

In the first case study, we consider a communication link with a total bandwidth of 12 units. We intend to support 3 different customer classes on that link. The parameters, which describe the customer classes, are indicated in Table 1. The discount rate was set to be 0.5, which translates into

Customer Class	Band-width, b_n	Arrival Rate, λ_n	Departure Rate, ν_n	Reward c_n
1	1	3	0.5	4
2	2	2	0.8	15
3	2	2.5	0.9	12

Table 1: Characterization of the different customer classes

Method	Average Reward	Lost Average Reward
Always Accept	40.1	31.8
TD(0)	45.6	26.3
Threshold Policy	47.2	24.7
Optimal Policy	47.2	24.7

Table 2: Performance comparison for the case with 3 different customer classes

a discount factor in the uniformized problem equal to 0.96.

As an approximation architecture, a multi-layer perceptron with one hidden layer and 3 hidden units was used.
Table 2 indicates the performance of the different policies. We see that the policy obtained through Neuro-Dynamic Programming is significantly better than the policy which tries to always accept a new customer. Simulation of the TD(0) policy revealed that only customers of class 1 are rejected. Based on this observation, we set the parameters for the Threshold policy. The obtained policy turned out to perform as well as the optimal policy obtained with Dynamic Programming.

5.2. Case Study with 10 Customer Classes

In the second case study, we consider a communication link with a total bandwidth of 600 units. We intend to support 10 different customer classes with that link. The parameters, which describe the customer classes, are indicated in Table 3. The discount rate was set to be 0.1, which translates into a discount factor in the uniformized problem of 0.9997.
As an approximation architecture, a multi-layer perceptron with one hidden layer and 10 hidden units was used.
Table 4 indicates the performance of the different policies. We see that the policy obtained through Neuro-Dynamic Programming is significantly better than the policy which tries to always accept a new customer. Simulation of the TD(0) policy revealed that only classes 6 and 8 ever get rejected. Again we used this information to set the parameters for the Threshold policy. Similarl to the first case study, we obtained a policy which performed better than the TD(0) policy.

Method	Average Reward	Lost Average Reward
Always Accept	388.5	35.5
TD(0)	394.3	29.7
Threshold Policy	395.8	28.2

Table 3: Performance comparison for the case with 10 different customer classes

Customer Class	Band-width. b_n	Arrival Rate, λ_n	Departure Rate, ν_n	Reward c_n
1	2	15	1.0	2
2	2	15	1.0	1.4
3	4	10	0.8	5
4	4	10	0.8	2.5
5	6	7	0.6	10
6	6	7	0.6	4
7	8	3	0.4	20
8	8	3	0.4	7
9	10	1.8	0.2	50
10	10	1.8	0.2	16

Table 4: Characterization of the different customer classes

6. CONCLUSIONS

Through this study, we showed that methods of Neuro-Dynamic Programming can provide a viable approach to the Admission Control Problem in an ATM environment. We have shown that the control policy obtained through Neuro-Dynamic Programming led to better policies than a policy that always accepts a new customer if the required bandwidth is available. Furthermore, based on simulations of the policy obtained with TD(0), we obtained an heuristic "Threshold" policy which performed as well as the optimal Dynamic Programming policy in the case of a communication link which supports 3 different customer classes.

7. REFERENCES

[1] D. P. Bertsekas, "*Dynamic Programming and Optimal Control*," Athena Scientific, 1995.

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

[3] G. J. Tesauro, "*Practical Issues in Temporal-Difference Learning*," Machine Learning, vol. 8, 1988.

[4] R. H. Crites and A. G. Barto, "*Improving Elevator Performance Using Reinforcement Learning* ," Advances in Neural Information Processing Sustems 8, MIT Press, 1996.

[5] W. Zhang and T. G. Dietterich, "*A Reinforcement Learning Approach to Job Scheduling*," Procceedings of the IJCAI, 1995

[6] S.P. Singh and D. P. Bertsekas, "*Reinforcement Learning for Dynamic Channel Allocation in Celluar Telephone Systems*," Neural Information Processing Systems Conference, 1996

[7] R. S. Sutton, "*Learning to Predict by the Methods of Temporal Differences*," Machine Learning, vol. 3, 1988