Machine Learning, 55, 5–29, 2004

A Reinforcement Learning Algorithm Based on Policy Iteration for Average Reward: Empirical Results with Yield Management and Convergence Analysis

ABHIJIT GOSAVI agosavi@buffalo.edu
Department of Industrial Engineering, The State University of New York at Buffalo, 342 Bell Hall Box 602050, Buffalo, NY 14260-2050, USA

Editor: Andrew Barto

Abstract. We present a Reinforcement Learning (RL) algorithm based on policy iteration for solving *average reward* Markov and semi-Markov decision problems. In the literature on *discounted reward* RL, algorithms based on policy iteration and actor-critic algorithms have appeared. Our algorithm is an asynchronous, *model-free* algorithm (which can be used on large-scale problems) that hinges on the idea of computing the value function of a given policy and searching over policy space. In the applied operations research community, RL has been used to derive good solutions to problems previously considered intractable. Hence in this paper, we have tested the proposed algorithm on a commercially significant case study related to a real-world problem from the airline industry. It focuses on yield management, which has been hailed as the key factor for generating profits in the airline industry. In the experiments conducted, we use our algorithm with a nearest-neighbor approach to tackle a large state space. We also present a convergence analysis of the algorithm via an ordinary differential equation method.

Keywords: reinforcement learning, average reward, policy iteration

1. Introduction

Markov decision problems (MDPs) are problems of decision making in which the decision maker has the objective of finding the optimal actions in the states visited by the system—that is, to maximize the value of some performance metric, such as long-run discounted reward or long-run average reward per unit time. Value iteration and policy iteration have remained as the methods of choice in designing Reinforcement Learning (RL) algorithms (see Sutton & Barto, 1996; Bertsekas and Tsitsiklis, 1996 for textbook treatment), especially when the decision-making problem has a Markovian or semi-Markovian nature.

For the discounted reward criterion, the most popular algorithm is Q-Learning, which closely follows value iteration in spirit. A policy iteration algorithm for discounted reward can be found in Rummery and Niranjan (1994) and Sutton (1996). An extension of Q-Learning, which was designed for MDPs, to discounted reward Semi-Markov Decision Problems (SMDPs) can be found in Bradtke and Duff (1995). See also Chapter 6, Section 6.2

of Bertsekas and Tsitsiklis (1996) for an account of "approximate policy iteration" algorithms for discounted reward.

In the average reward case, algorithms such as R-Learning (see Schwartz, 1993), SMART (see Das et al., 1999), Relaxed-SMART (see Gosavi, 1999, 2003), and the algorithm in Abounadi, Bertsekas and Borkar (1998) have been proposed. All these algorithms use some form of value iteration. In SMART and Relaxed-SMART, a form of the average reward Bellman equation is directly used while the algorithm in Abounadi, Bertsekas, and Borkar uses a distinct updating scheme based on a form of value iteration given in Bertsekas (1995b) that has been proven to converge. Relaxed-SMART has been proven to converge in Gosavi (2003). Relaxed-SMART and SMART work for both MDPs and SMDPs. Konda and Borkar (1999) have presented an actor-critic algorithm for average reward.

In the RL literature, an algorithm based on policy iteration seems to be lacking for the average reward case. In this paper, we discuss some of the difficulties in solving average reward SMDPs and present a model-free, average-reward, asynchronous RL algorithm that is based on policy iteration. The algorithm can be used to solve both MDPs and SMDPs.

It must be emphasized that the updating equations in the algorithm proposed here are different from those in Das et al. (1999), Gosavi (2003), or Abounadi, Bertsekas, and Borkar (1998). They are directly derived from policy iteration in classical dynamic programming. Also, we must point out that the work of Konda and Borkar (1999), which follows policy iteration *in spirit*, uses a totally different (although convergent) mechanism that has roots in the classical actor-critic framework (see Sutton & Barto, 1996).

We also present a convergence analysis using an ordinary differential equation method. Finally, we reinforce the theoretical convergence analysis with numerical experiments on a real-life problem with a large state space. The selected problem domain is one from the field of operations research, where RL has been used to solve difficult problems. The problem studied in this paper is related to yield management, which has been hailed as the single most important contributing factor to the success of airline industries in the last two decades (see Davis, 1994 and Horner, 2000).

The SMDP is more general than the MDP because while calculating a given performance metric (e.g., average reward and discounted reward), it takes into account the time spent by the system in each state transition of the Markov chain. In the real world, more problems tend to be SMDPs than MDPs because profits and costs are invariably tied to some time frame.

Also, the long-run average reward criterion is an important metric in its own right. Some of the pioneering work in this area can be attributed to Schwartz (1993) and Mahadevan (1994, 1996a, 1996b). However, regular *value iteration* is not guaranteed to generate optimal solutions in average reward SMDPs, which is the prime motivation for developing an RL algorithm based on *policy iteration*. We shall explore this issue in some detail in this paper.

The rest of this paper is organized as follows. We begin with a discussion of the Markov decision framework and RL algorithms in Section 2. We also discuss the need for an approach based on policy iteration in the case of SMDPs in the average reward case. In Section 3, we present our algorithm. In Section 4, we present the airline case study. In Section 5, we present a convergence analysis of the new algorithm. Section 6 contains some concluding remarks and some suggested directions for further research in this area.

2. The Markov and semi-Markov decision framework

We shall concentrate in this paper on MDPs and SMDPs with a finite state space *and* a finite action space. For a detailed discussion on MDPs and SMDPs, the reader is referred to Puterman (1994). In what follows, we present a quick description of the notation that we shall use in the rest of the paper. S will denote the finite state space of the problem, $A(i)$ will represent the finite set of actions allowed in state i, and $r(i, a, j)$, $p(i, a, j)$, and $t(i, a, j)$ will denote, respectively, the following three quantities associated with a transition from state i to state j under the influence of action a: the immediate reward earned in the transition, the probability of the transition, and the time spent in the transition.

Average reward for a unichain SMDP (see Puterman (1994) for a definition), starting from state i and following policy π, can be mathematically expressed as:

$$\rho(i) = \liminf_{T\to\infty} \frac{E\left[\sum_{k=1}^{T} r(x_k, \pi(x_k), x_{k+1}) \mid x_1 = i\right]}{E\left[\sum_{k=1}^{T} t(x_k, \pi(x_k), x_{k+1}) \mid x_1 = i\right]}$$

where x_k is the state from which the kth jump of the Markov chain occurs and $\pi(x_k)$ denotes the action selected in state x_k when policy π is followed. It can be shown that the average reward is independent of the starting state for unichain Markov chains. Also, an expression for the average reward—for the case of MDPs—can be obtained from the above by setting $t(i, a, j) = 1$ for all possible values of i, j, and a.

Dynamic programming (DP) is a powerful tool that can be used to solve MDPs and SMDPs. Traditional DP techniques require the values of all the transition probabilities, the knowledge of which is often very hard to obtain especially if (1) the system has a very large state space and/or (2) the dynamics of the system are governed by complex transition mechanisms (the complexity often depends on the number of random variables involved in the system and the role they play in the dynamics of the system). Under such circumstances, RL, which is a relatively new methodology, can be used. The so-called model-free (as opposed to model-based) algorithms of RL can solve MDPs without calculating the transition probabilities. In many SMDPs, the transition probabilities may be available but the same may not be true of transition times. Here too model-free RL can be used for solution purposes. We next present an important and well-known theorem, which holds the key to a DP solution to an SMDP.

Theorem 1. *Given that the objective is to maximize the long-run average reward calculated over an infinite time horizon for any finite-state unichain SMDP, there exists a scalar ρ^* and a value function $R^*(\cdot)$ satisfying the following system of equations for all $i \in S$,*

$$R^*(i) = \max_{a\in A_i} \left(\bar{r}(i, a) - \rho^* y(i, a) + \sum_{j\in S} p(i, a, j) R^*(j) \right), \tag{1}$$

such that the greedy policy π^* formed by selecting actions that maximize the right-hand side of the above equation is average reward optimal, *where $\bar{r}(i, a)$ is the expected immediate reward when an action a is taken in state i, $y(i, a)$ is the expected transition time in state i*

when action a is taken in state i, and $p(i, a, j)$ is the probability of transition from state i to state j under action a in one step.

Note, that by their definitions, $\bar{r}(i, a) = \sum_j p(i, a, j)r(i, a, j)$ and $y(i, a) = \sum_j p(i, a, j)t(i, a, j)$. Also note that the scalar ρ^*, described in the statement of the theorem, can be shown to be the maximum average reward that can be obtained from the SMDP.

2.1. *Value iteration for MDPs*

Theorem 1 provides us with a method to find the optimal solution to an SMDP. It is to be noted that Theorem 1 reduces to the same result for the MDP case when $y(i, a)$ is set to 1. In a method of successive approximations, we generally start with an arbitrary value for the unknown vector R and use Eq. (1) repeatedly, as a transformation, to obtain successive approximations of the vector R^*. The transformation can be expressed as:

$$R^*(i) \leftarrow \max_{a \in A_i} \left(\bar{r}(i, a) - \rho^* y(i, a) + \sum_{j \in S} p(i, a, j) R^*(j) \right) \quad \text{for all } i. \tag{2}$$

Transformation (2) cannot be used directly because it involves ρ^*, the optimal average reward, which is unknown. In the MDP case, where $y(i, a) = 1$, if we set ρ^* to 0, we have what is called regular value iteration. Next, consider the transformation in value iteration:

$$R^*(i) \leftarrow \max_{a \in A_i} \left(\bar{r}(i, a) + \sum_{j \in S} p(i, a, j) R^*(j) \right). \tag{3}$$

If $y(i, a)$ equals 1, the vector of values generated by transformation (2) and the vector generated by transformation (3) would differ by a constant in every iteration, but would yield the same policy. Transformation (2) is known to yield the optimal solution, i.e., produce the maximum average reward. Hence value iteration, that is, transformation (3) can be used for MDPs.

However, in the SMDP case, it is not mathematically correct to replace ρ^* by 0, since the transformation obtained with such as replacement is not necessarily equivalent to transformation (2). For instance, consider an SMDP with two states and two actions in each state in which $y(1, 1)$, $y(1, 2)$, $y(2, 1)$, and $y(2, 2)$ are all unequal. Then the values, for any given state i, generated from one application of (2) and the values generated from one application of (3), will *not* differ by a constant. So, (3) and (2) will lead to *different* policies. Hence (2), which is guaranteed to give an optimal solution, cannot be substituted for (3). For this reason, an approximate version of value iteration has been suggested in the literature, which we discuss next.

2.2. *Value iteration for SMDPs*

In value iteration for SMDPs, one has to "uniformize" the SMDP—thereby converting the SMDP into an MDP. See Puterman (1994) and Bertsekas (1995a) for detailed discussions on this conversion process. The updating equation for the uniformized problem, which may be used in value iteration, is as follows. For all $i \in S$,

$$R^*(i) \leftarrow \max_{a \in A_i} \left\{ \bar{r}^t(i, a) + \sum_{j \in S} p^t(i, a, j) R^*(j) \right\},$$

where the replacements for $\bar{r}(i, a)$ and $p(i, a, j)$ are denoted by $\bar{r}^t(i, a)$ and $p^t(i, a, j)$ respectively and defined as:

$$\bar{r}^t(i, a) = \bar{r}(i, a)/y(i, a),$$

$$p^t(i, a, j) = \eta p(i, a, j)/y(i, a), \quad \text{if } i \neq j,$$

and

$$p^t(i, a, j) = 1 + \eta[p(i, a, j) - 1]/y(i, a), \quad \text{if } i = j.$$

In the above, η should satisfy

$$0 \leq \eta \leq y(i, a)/\{1 - p(i, a, i)\}$$

for all a, i and j. It is clear that to be able to carry out such a "uniformization," one needs to know the transition probabilities. Consequently, an approach of this type is not very suitable for a model-free RL implementation, which seeks to obtain a solution without attempting to find the transition probabilities. The other way out of this difficulty is to use an approximation of the optimal value of the average reward and use a relaxed version of transformation (2). This is the central idea underlying SMART-based algorithms that rely on the Bellman equation.

2.3. Policy iteration

Since value iteration for SMDPs involves a cumbersome transformation, it is only natural to look towards the other technique of dynamic programming—policy iteration—for solution. Fortunately, policy iteration for SMDPs is *exact* and neither requires prior knowledge of the optimal average reward, nor does it require any "uniformization." The idea in policy iteration is to start with a randomly chosen policy, compute the value function associated with it, and then use it to find an improved policy. The process continues until no further improvement is possible. For algorithmic details of policy iteration, the reader is referred to Puterman (1994). Policy iteration is a *clean* way of solving the SMDP since *no "uniformization" is required.* As mentioned earlier, uniformization may be difficult when the transition probabilities are not available.

3. RL

The underlying idea in most RL schemes is to approximate the value function (either the optimal value function or the value function associated with a given policy) using stochastic approximation. In *policy iteration*, the value function associated with any *given policy* has to be computed in every iteration, while in value iteration, the goal is to approximate the value function associated with the *optimal policy*.

In RL, the optimal policy is determined via a reinforcement mechanism. In such a setting, the decision maker can be viewed as an *agent* that selects actions in each state visited either on a real-time basis or in a simulator. The *feedback* obtained from every action selected is used to update the *knowledge base* of the agent. The knowledge base of the agent, usually, contains the so-called Q-factors. By trial and error, with repeated reinforcements, the agent learns the optimal policy. The book of Sutton and Barto (1996) describes this methodology in great detail.

The Q-factor we are about to present is based on policy iteration and is defined in terms of a given policy. For a state-action pair (i, a) for a policy π where $a \in A(i)$:

$$Q_\pi(i, a) = \sum_{j \in S} p(i, a, j)[r(i, a, j) - \rho_\pi t(i, a, j) + R_\pi(j)], \tag{4}$$

where ρ_π and R_π are respectively the average reward and the value function associated with the policy π. Using this definition, we can come up with a version of policy iteration in terms of Q-factors. We shall present this version because the new algorithm presented in this paper follows from it in a straightforward manner.

3.1. *Q-factor based policy iteration for SMDPs under average reward*

Now from the Bellman equation, we have that for a given policy π:

$$R_\pi(i) = \sum_{j} p(i, \pi(i), j)[r(i, \pi(i), j) - \rho_\pi t(i, \pi(i), j) + R_\pi(j)], \quad \text{for all } i. \tag{5}$$

From Eqs. (5) and (4), we have that

$$R_\pi(i) = Q_\pi(i, \pi(i)), \quad \text{for all } i. \tag{6}$$

Using Eq. (6), Eq. (4) can be written as:

$$Q_\pi(i, a) = \sum_{j \in S} p(i, a, j)[r(i, a, j) - \rho_\pi t(i, a, j) + Q_\pi(j, \pi(j))], \quad \text{for all } i \in S \text{ and } a \in U(i). \tag{7}$$

Equation (7) is a Q-factor version of the equation used in policy iteration for policy evaluation and is thus useful in devising a Q-factor version of policy iteration, which we present next.

Step 1: Set $k = 1$. Select a policy arbitrarily and denote it by π_k. Let n be the number of decision-making states.

Step 2: For the policy, π_k, obtain the values of Q_{π_k} by solving the following system of linear equations:

$$Q_{\pi_k}(i, a) = \sum_{j=1}^{n} p(i, a, j)[r(i, a, j) - \rho_{\pi_k} t(i, a, j) + Q_{\pi_k}(j, \pi_k(j))]$$

for each $a \in U(i)$ and each $i \in S$. The unknowns are the Q-factors and ρ_{π_k}. The linear system above has one unknown more than the number of equation; the extra unknown is ρ_{π_k}. Solve it by setting any one Q-factor to 0.

Step 3: Generate a new policy π_{k+1}, using the following relation:

$$\pi_{k+1}(i) = \arg\max_{u \in U(i)} Q_{\pi_k}(i, u).$$

Step 4: If the policy π_{k+1} is identical to policy π_k, stop; else set $k \leftarrow k + 1$ and go back to Step 2.

In the next section, we describe our RL algorithm in detail.

4. *Q*-*P*-Learning

Since the algorithm that we will develop in this section closely follows the spirit of policy iteration, we will next describe it in terms of policy iteration as follows. Begin by selecting a policy arbitrarily and estimate (using simulation) the average reward associated with it. Thereafter, perform policy evaluation; that is, use an updating transformation based on (7) in a simulator to evaluate the Q-factors associated with that policy. We refer to each policy-evaluation step as a *phase* in the learning process in which the Q-factors of a particular policy are learned.

What is done during a phase is essentially equivalent to the policy evaluation stage of policy iteration. Then we go on to perform policy improvement, where we select our new policy based on the Q-factors just generated. Before beginning a phase, we must first obtain the average reward associated with the new policy (which can be done using simulation). The old vector of Q-factors will now be called P-factors, (P for Policy), since these P-factors will contain information about the policy which will be evaluated in the next phase. Fresh Q-factors will be approximated in the next phase. Hence the name Q-P-Learning. The step-by step details are in figure 1.

Some additional comments are in order in relation to figure 1.

Comment 1. The learning rate β needs to be decayed with the iteration. We can choose $\beta = 1/m$. The value of $m_{\max}$ should increase with the number of phases. A rule such $m_{\max} = 1000 + E^2$ can be used.

```
ALGORITHM $Q$-$P$-Learning:
Step 1: Initialize the vector $P(i,a)$ for all states $i \in S$ and all $a \in U(i)$ (the set of permissible actions in state $i$) to random values. Set $E = 1$ (where $E$ denotes the number of phases) and initialize $E_{\max}$, $m_{\max}$, $h_{\max}$, and $T_{\max}$ to large numbers.
Step 2: Set $Q(i,a) = 0$ for all $i \in S$ and all $a \in U(i)$. Simulate the system for a time interval of $T_{\max}$, using the action given by $\arg\max_{b \in U(i)} P(i,b)$ in every $i \in S$. At the end of the simulation, divide the total of immediate rewards by $T_{\max}$ to obtain an estimate of the average reward $\rho$. Use averaging with $h_{\max}$ replications to obtain a good estimate of $\rho$. Set $m$, the iteration number within a phase, to 0.
Step 3: (Policy evaluation) Start fresh simulation. Let the system state be $i \in S$.
Step 3a: Simulate action $u \in U(i)$ with probability $1/|U(i)|$.
Step 3b: Let the next decision-making state encountered in the simulator be $j$. Also, let $t(i,u,j)$ be the transition time (from state $i$ to state $j$) and let $r(i,u,j)$ be the immediate reward.
Step 3c: Calculate $\beta$ using $m$ (see Comment 1 below). Then change $Q(i,u)$ using:
    $Q(i,u) \leftarrow (1-\beta)Q(i,u) + \beta[r(i,u,j) - \rho t(i,u,j) + Q(j, \arg\max_{v \in U(j)} P(j,v))]$.
Step 3 d: Increment $m$ by 1. If $m < m_{\max}$ set current state $i$ to new state $j$ and then go to Step 3a; else go to Step 4.
Step 4: (Q to P conversion - policy improvement) Set $P(i,a) \leftarrow Q(i,a)$ for all $i$ and $a \in U(i)$. Set $E \leftarrow E + 1$. If $E$ equals $E_{\max}$ terminate learning; else go back to Step 2.
```

Figure 1. A policy iteration based RL algorithm for computing average reward optimal policies for MDPs and SMDPs.

Comment 2. Each action is chosen in each state with equal probability so that all actions are tried sufficiently often in each state. This ensures that the simulation-based estimates of the Q-factors converge to their correct values.

Comment 3. The algorithm described above uses value iteration in the policy evaluation stage. This idea can be found in SARSA (see Sutton & Barto, 1996, pp. 145–148) and in the DP literature in the modified policy iteration algorithm of Puterman and Shin (see Puterman, 1994). Section 6.2 of Bertsekas and Tsitsiklis (1996) discusses a method that employs the same philosophy. They have called it "approximate (non-optimistic) policy iteration" and have established bounds on its worst case behavior. Their analysis is performed for the discounted problem and the stochastic shortest path problem. In particular they show that their method is bound to converge in the presence of simulation noise *and* function approximation noise, and that the policy generated by the policy update cannot be much worse than the previous policy.

5. Case study

The deregulation of the airline industry in 1978 allowed airline companies to choose their own market segments and routes and set their own fares as long as they complied with the safety and security regulations enforced by the Federal Aviation Administration (FAA). With the fierce competition that ensued in the airline industry as a result of this, it was soon realized that optimization techniques were needed for optimal use of the expensive resources of these industries. American Airlines took a lead but all major air-carriers now use some kind of optimization in controlling their operations. It is widely perceived that unless one uses these techniques there is a strong possibility of losing market-share and going out of business because one's competition will put them to good use. The most important goal of any airline company is that of maximizing the revenue obtained from the sale of tickets on any given flight. This is referred to in the literature as *yield management*. It forms a crucial aspect of airline logistics. According to a report from SIAM news (see Davis, 1994): "Yield management... saved American Airlines \$1.4 billion in the period from 1989 to 1992, about 50% more than its net profit of \$892 million for the same period. Modeling and optimization made the difference between profit and loss." And according to a report (Horner, 2000) "By 1998,... the revenue management system at American Airlines was generating nearly \$1 billion in annual incremental revenue. To put that figure into perspective, consider that the airline's total operating profit didn't approach \$1 billion until 1997."

Seat allocation and overbooking are two of the most important aspects of the yield management problem, and the key to making profit lies in exercising control over the seat allocation and the overbooking aspects. The problem considered here is a combined problem of seat allocation and overbooking having the following features: a single flight leg with multiple fare classes, concurrent and random demand arrivals from the different fare classes, class-dependent and random cancellations. Accommodating all these features is computationally challenging because that makes it a large-scale and complex stochastic optimization problem. Most models in the existing literature accommodate only a subset of these features to make the model tractable. At the same time, because of the high stakes involved, there is a need for an approach that accounts for all these factors and generates optimal or near-optimal results. In fact, according to a recent review paper on revenue management written by McGill and van Ryzin (1999): "Recently developed approximation methods for dynamic programming and stochastic programming (RL) may be useful in revenue management." Hence RL obviously seems to have the potential for offering solutions here. We use RL, using Q-P-Learning, on this problem accounting for all the factors mentioned above. For an alternative RL approach to this problem (see Gosavi, Bandla & Das, 2002).

The seat allocation problem has its roots in the airlines' practice of selling seats within the same cabin of a flight at different prices. There are differences in customers, and airlines use these differences to sell tickets at different prices. A passenger who wants a lesser number of stopovers or a passenger who arrives late in the booking process is made to pay a higher fare. And this results in classifying passengers into different fare classes based on their needs and the circumstances. A clever thing to do is to protect some seats from the lower revenue fare classes in order to be able to satisfy demands from the higher revenue classes since that helps in increasing the revenue. But then the obvious question that arises is: what is the optimal level of protection?

The "overbooking" aspect adds a new dimension to this problem. It arises from the fact that customers do cancel their tickets and often fail to show up at the flight time (no-shows).

Airlines tend to "overbook" fights (sell more tickets than the number of seats available) anticipating such events to avoid flying with empty seats. It may be noted that a seat in an airplane (like a hotel room or vegetables in a supermarket) is a perishable item; its value becomes null as soon as the flight takes off. However, associated with excessive overbooking is the risk of not having enough seats for all the ticket holders. When this happens, airlines deny (bump) boarding request to the extra ticket- holders and pay a penalty in two ways: directly in the form of monetary compensations to the inconvenienced passengers, and indirectly via loss of customer goodwill. Hence, a prudent choice of an overbooking policy that maximizes the revenue is called for.

A primary factor used in classification is the time of the request for reservation. Passengers who book in advance get discount fares; in other words they belong to lower fare classes. Those who come later have to pay higher fares. If classification is carried out on the basis of this factor alone, it is adequate to assume that passengers in different fare classes arrive sequentially, i.e., passengers in the lowest classes arrive first, followed by passengers in the next higher class and so on. However, another factor is also used in classification, which is that of the itinerary of the passenger. To see how an origin-and-destination based (itinerary-based or number of stop-overs based) passenger classification works, consider a single flight leg from Dallas to New York. The flight in addition to carrying passengers whose origin is Dallas and destination is New York, is likely to be carrying passengers from other longer itineraries, such as San Juan-Miami-Dallas-New York, Phoenix-Dallas-New York, and Tampa-Dallas-New York. If the passengers whose itinerary originates from Dallas form the highest class, those flying from Miami (or Tampa) to New York via Dallas form the middle class, and those flying from San Juan to New York via Miami and Dallas (and are on the third leg of their itinerary) form the lowest fare class of passengers in the plane. See also figure 2, which explains this idea. A circle in figure 2 represents the origin and the symbol inside it indicates the class of the passenger originating at that point. The figure also shows that usually itinerary-based classification yields two or three fare classes. It may be noted that if this happens to be one of the factors used to classify passengers, sequential passenger arrival will be a poor assumption and as suggested by Robinson (1995), a *concurrent* arrival pattern must be assumed. Passenger classification in the real world is made on the basis of several factors. We do not concern ourselves with how airlines classify passengers because the model presented in our paper can be used for any method of passenger classification.

5.1. Literature overview

The process of maximizing revenue subject to constraints, such as the limits on the number of fare classes available and the number of seats available in each class, was studied in Belobaba (1989). This work used a generalized (multiperiod) version of the equation in Littlewood (1972) to obtain allocations for more than two fare classes. Belobaba's method, named the EMSR (Expected Marginal Seat Revenue) model, assigns a fixed quantity of seats to each fare class. Brumelle and McGill (1993), Curry (1990), van Ryzin and McGill (2000), and Wollmer (1992) reported some shortcomings of the EMSR model. Glover et al. (1982), Chatwin (1998), and Shlifer and Vardi (1975) are some of the other researchers who have done significant work in this area. Howard (1971) cast the airline overbooking

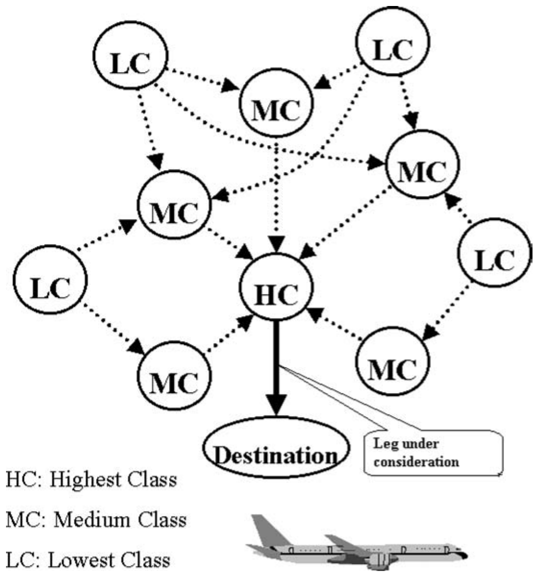

Figure 2. A schematic showing classification of customers based on the origin (circle) and the destination, in one particular leg of an airline flight.

problem for a single fare class as an MDP and proposed the use of value iteration; however, only very small problems can be solved with this approach because of its computational inadequacies. Subramaniam, Stidham and Lautenbacher (1999) also model the problem as an MDP by discretizing the time horizon and using a uniformization technique to convert the SMDP to an MDP. They are able to make the problem tractable by assuming that the *cancellation probability is independent of fare class*. We do not make this assumption and we do not discretize the time. We retain the SMDP characteristics and consider most of the complexities mentioned above. We then use Q-P-Learning to generate a solution for this SMDP.

5.2. SMDP model

To model the yield-management problem as an SMDP, we first define the system state space. Conservatively, for n fare classes, the system state (ϕ) can be denoted by the vector

$$\phi = (c, s_1, s_2, \ldots, s_n, \psi_1, \psi_2, \ldots, \psi_n, t),$$

where c represents the class of the most recent customer (among n possible classes) seeking a ticket, $s_1, s_2, \ldots, s_n$ represent the number of seats sold in n classes, ψ_k, for $k \in \{1, 2, \ldots, n\}$, is a vector of size s_k containing the times of arrivals of all the customers with a class k ticket, and t represents the time remaining to departure of the flight. The cardinality of the finite part of the state space can be shown to have an upper bound of $n \cdot M^n$, where M is the maximum number of customers in any given class with tickets for the plane. Note that for all practical purposes, M could be equal to the total flight capacity plus the overbooking amount.

Clearly, a change in the system state is caused by any one of the following three events: (1) a new customer arrives to the system requesting a ticket, (2) a cancellation occurs and (3) the flight departs. Whenever a customer places a request for a seat, a decision needs to be made regarding whether to accept or deny the request. The conservative state-space definition given above satisfies the Semi-Markov property described in Section 2. The action space, which is common to all the decision-making states is, $A = \{accept, deny\}$. It may be noted that the decision-maker can address both the seat-allocation problem and the overbooking aspects of the yield management problem by selecting a policy vector (π) of the size of the state space that contains either "accept" or "deny" decision for every state visited by the system. The metric used in this paper is long-run average reward.

5.2.1. Why average reward? The performance metric chosen to be used for the yield management problem is average reward although the problem can be studied as a finite horizon problem. If studied as a finite horizon problem, the problem would become a stochastic shortest path problem. The average reward problem has been shown to be mathematically equivalent to the stochastic shortest path problem (see Bertsekas, 1995a). The average reward problem can also be viewed as a discounted reward problem in which the discounting factor tends to 1. However, it has been shown in Puterman (1994) on page 165 (in the first paragraph of Chap. 8) that the convergence *rates* can become very small as the discounting factor approaches 1. As a result, the theory of average reward algorithms should be developed separately. On page 331, Puterman writes that "the average reward criterion occupies a cornerstone of queuing control theory especially when applied to computer systems and communications networks... the criterion may also be appropriate for inventory systems with frequent restocking decisions."

5.3. EMSR heuristic

In what follows, we present a heuristic algorithm to benchmark the performance of our algorithm. The heuristic that we present is the most widely used heuristic in the industry. Belobaba (1989) developed the EMSR (Expected Marginal Seat Revenue) model using an equation in Littlewood (1972). We refer to the EMSR model of Belobaba as the EMSR (Expected Marginal Seat Revenue) heuristic. The strategy adopted in the EMSR heuristic consists of determining *booking limits* for the different fare classes (using Littlewood's equation). When a customer belonging to a particular class requests a ticket in a particular class, he/she is given a reservation only if the number of seats sold in that class is less than the booking limit for that class.

To obtain the booking limits for any given class, Littlewood's equation is solved to obtain the unknown quantities S_j^i for $i > j$, the equation being:

$$P\big(X_i > S_j^i\big) = f_j/f_i \quad j = 1, \ldots, k, \tag{8}$$

where X_i denotes the number of requests for class i that will arrive in the booking horizon, S_j^i denotes the number of seats to be protected from class j for higher class i, f_i and f_j

are the fares for the classes i and j respectively, and k is the number of classes. Once the quantities S_j^i are obtained, the booking limit for a class j may be obtained by:

$$L_j = C - \sum_{i>j} S_j^i, \tag{9}$$

where C is the capacity of the plane. Cancellations and no-shows are incorporated in the EMSR heuristic by multiplying the capacity of the aircraft by an overbooking factor. Thus, if C is the capacity of the flight and p is the probability of cancellation, then the overbooking factor is given by $1/(1-p)$ and the modified capacity of the aircraft is given by $C/(1-p)$.

A strong merit of Littlewood's approach is that all that is needed for it is the solution of a simple linear equation. If no overbooking or cancellations are considered, Littlewood's solution gives near optimal results. However, when we consider additional modeling assumptions such as overbooking and cancellations, Littlewood's approach starts deviating from optimality. Under these assumptions, it becomes necessary to use more powerful models such as the MDP and the SMDP.

5.4. *Results*

In this section, we describe the sample problems selected for numerical analysis of *Q-P*-Learning. We first discuss some of the implementational issues of function approximation.

5.4.1. Function approximation. The conservative definition of the state space makes it uncountably infinite. We hence need a function approximation scheme that helps us encode the state space keeping its important features intact. Finding a suitable function encoder is a challenging task and the key to finding it frequently lies in the problem structure. After considerable experimentation, we found a scheme that relates to the revenue earned in a flight and captures the salient parts of the state space adequately (the problem is after all a revenue management problem). We shall describe the scheme below.

We denote our state space by $\{c, PI\}$ where PI is defined below and depends on the number of seats sold in each class, while c denotes the class of the current customer. It is clear that acceptance and rejection of a passenger have a direct bearing on the class of the customer and how much revenue has already been earned in that flight. Hence to quantify the dollar value in the current number of bookings, we define a "payload index" (PI) as:

$$PI = \frac{\sum_{i=1}^{N} Fa_i SS_i}{D}. \tag{10}$$

In the above equation, Fa_i denotes the fare of the ith class, SS_i denotes the seats sold in the ith class and N denotes the number of classes. D denotes a factor that simply reduces the range of the numerator. We have a continuous state space here since PI is a continuous variable. We handle it using a nearest-neighbor approach using the following rule. The state corresponding to the nearest integer to the actual value of PI is updated. Thus we actually update only a finite set of values. The payload index, it is to be understood, is proportional to the total revenue of the company so far and consequently keeps changing with time. It also retains the dynamic nature of the actual state space.

5.4.2. Experiments. In order to evaluate the performance of Q-P-Learning, we studied a set of sample seat allocation problems with the following characteristics.

1. The flight has three fare classes and the fare structure is represented by the vector $FS = (f_1, f_2, f_3, b)$, where f_i is the fare of the ith class, and b is the bumping cost. A lower value of i stands for a lower revenue fare class. Two different sets of fare structures—FS_1 and FS_2—were considered.
2. The customer arrival process is Poisson with parameter λ.
3. The capacity of the aircraft is C.
4. The customer distribution for class $i \in \{1, 2, 3\}$ is represented by the vector $CD = (p_1, p_2, p_3)$, where p_i is the probability of a customer belonging to the ith class. Clearly, $p_1 + p_2 + p_3 = 1$.
5. Every customer belonging to class i has a probability of p_c^i of canceling the trip and the time of cancellation is uniformly distributed between the time of sale and the time of flight departure.
6. A customer who decides to cancel his/her ticket pays a penalty depending on the time remaining for flight departure. So if he/she has paid a fare of F dollars, and the time remaining is t in the horizon of length H, then the penalty paid $= F(H - t)/H$. The penalty calculation depends on the airline policies, which vary with the airline and even the time of the year. The formula used above is not the most general model for penalty calculation; however, any model can be easily used in a simulator.

Table 1 summarizes all the numerical problems that were examined using the P-Q-Learning. The results were averaged over 10 runs, where each simulation run lasted for 1 million time units (several thousand flights). In Table 1, $\rho_{Q\text{-}P\text{-Learning}}$, $\rho_{Q\text{-Learning}}$, and ρ_{EMSR} denote respectively the revenue generated (in \$) per unit time by using the Q-P-Learning policy,

Table 1. Sample problems showing a comparison of the average revenue using EMSR, Q-P-Learning, and Q-Learning. Im denotes the improvement of Q-P-Learning over EMSR.

$FS_1 = (100, 175, 250, 300)$, $FS_2 = (199, 275, 350, 400)$
$CD = (0.7, 0.2, 0.1)$, $\lambda = 1.4$, Horizon = 100 days, Capacity = 100 seats
$p_c^1 = (0.1, 0.1, 0.1)$, $p_c^2 = (0.1, 0.2, 0.3)$, $p_c^3 = (0.15, 0.2, 0.25)$

Case	FS_i	p_c^i	D	ρ_{EMSR}	$\rho_{Q\text{-}P\text{-Learning}}$	Im (%)	$\rho_{Q\text{-Learning}}$
1	FS_1	p_c^1	500	134.43 ± 0.74	138.8 ± 0.63	3.2	135.67 ± 0.45
2	FS_2	p_c^1	1000	246.61 ± 1.58	257.62 ± 1.51	4.5	250.12 ± 0.12
3	FS_1	p_c^2	500	125.64 ± 0.87	141.61 ± 0.64	12.7	135.45 ± 0.23
4	FS_2	p_c^2	1000	226.28 ± 1.81	262.51 ± 0.96	16	240.11 ± 0.68
5	FS_1	p_c^3	500	131.00 ± 1.63	144.58 ± 0.78	10.4	134.78 ± 0.19
6	FS_2	p_c^3	1000	236.67 ± 3.43	266.47 ± 0.86	12.6	246.10 ± 0.76

the Q-Learning policy, and the EMSR heuristic policy. The 95% confidence intervals on the revenues are also shown. The *countable* part of the state space of the largest problem considered here is of the order of 10^9 and improvement over EMSR of up to 16 percent is seen. Q-Learning has done reasonably well against EMSR. In comparison to our algorithm, however, Q-Learning has not done very well, which can be easily explained. Q-Learning is not designed for average reward problems, and that it is able to outperform EMSR speaks for its robustness. Q-learning was done with the discounting factor set to 0.99.

6. Convergence analysis

Two major issues are involved in showing the convergence of Q-P-Learning. First, it has to be established that Step 3 can be successfully completed in a finite number of iterations. Secondly, it has to be shown that this occurs in the asynchronous conditions of the simulator.

Two approaches to show convergence in Step 3, in the presence of noise, are the "reducing cube-size" method of Bertsekas and Tsitsiklis (1996) and the Ordinary Differential Equation method of Kushner and Clark (1978). We shall show convergence using the latter, although it can also be done via the former. Convergence under asynchronous conditions follows from Borker (1998). In the next section, we shall prove the convergence of Q-P-Learning. This convergence result appears as Theorem 3. We next state an important result from Kushner and Clark (1978) that forms the starting point of the proof for Theorem 3.

Theorem 2. *Consider a stochastic approximation scheme given by*:

$$x_l^{k+1} = x_l^k + a^k\big(f\big(x_l^k\big) + w^k\big), \quad l = 1, 2, \ldots, n, \tag{11}$$

where x_l^k denotes the lth element of the iterate in the kth iteration, a^k denotes the learning rate at the kth iteration, f denotes the function on the vector x^k from R^n to R^n, and w^k denotes a noise term in the kth iteration.

Assume the following conditions to hold:

Condition 1. The function f is Lipschitz continuous.

Condition 2. The learning rate a^k satisfies the following conditions:

$$\sum_{k=0}^{\infty} a^k = \infty \quad \textit{and} \quad \sum_{k=0}^{\infty} (a^k)^2 < \infty.$$

Condition 3.

$$E[w^k \mid x^0, x^1, \ldots, x^k, w^0, w^1, \ldots, w^{k-1}] = 0,$$

$$E[\|w^k\|^2 \mid x^0, x^1, \ldots, x^k, w^0, w^1, \ldots, w^{k-1}] <= A + B\|x^k\|^2,$$

for some constants A and B, and where $\|.\|$ indicates some norm.

Condition 4. The iterates (x^k) are bounded, and

Condition 5. The ordinary differential equation (*ODE*)

$$\frac{dx(t)}{dt} = f(x(t))$$

has a unique globally asymptotically stable equilibrium point x^*,

then the sequence $\{x^k\}$ *converges to* x^*.

Remark 1. The iterate in the theorem presented above could be the Q-factor with l denoting a state-action pair.

Remark 2. The second condition contains standard rules for learning rates, without which "averaging"—that helps us avoid having to compute the transition probabilities—cannot be achieved. Essentially, the third condition says that the conditional mean of the noise term is 0, and its conditional variance is finite.

Remark 3. All the result given above says is that the sequence of iterates (Q-factors in our case) will converge to a fixed point.

Remark 4. The theorem above can be shown to hold for asynchronous iterates under the additional assumption (see Borkar, 1998) that there exists some $C > 0$ such that:

$$\liminf_{k\to\infty} \frac{q(k,l)}{k+1} \geq C,$$

where $q(k,l)$ is the number of times the state-action pair l has been tried till the kth step in the learning process. This condition ensures that the difference between the number of times an iterate has been updated and the same number for any other iterate is a finite number. The ODE analysis in Borkar (1998) shows that even under asynchronous conditions, there exists a similar ODE, and convergence is guaranteed under asynchronous conditions as well.

We are now at a position, where with some additional work, we can use the results given above to prove that our algorithm will converge to an optimal solution. We next state our main result.

Theorem 3. *The algorithm described in Section* 3 *generates an optimal solution to the SMDP in question under the following assumption*:

Assumption A: There exists a state $s \in S$, where S is the set of all states in the underlying Markov chain, such that for some integer $m > 0$, and for all initial states and for all stationary policies, s is visited with positive probability at least once in the first m jumps of the Markov chain.

Assumption A restricts the above result to problems with at least one recurrent state. The Q-factors of that state may have to be fixed to some value.

Proof of Theorem 3: To prove this result, we need to establish two facts: (1) the Q-factors generated in the policy evaluation stage of our algorithm, after a finite number of iterations, reach values which enable us to perform Step 4 (policy improvement step) of the algorithm correctly—that is, enable us to select the same policy in the policy improvement step that would be selected by the limiting values of the Q-factors of Step 3. (2) The estimate of ρ in Step 2 does not affect the algorithm's path. Now, in all other respects, the algorithm mimics policy iteration, which is proven to converge Puterman (1994).

The core of the policy evaluation stage can be expressed by the following transformation:

$$\begin{aligned} Q^{k+1}(i,a) \leftarrow\ & Q^k(i,a) + \beta\big[r\big(i,a,e^k_{i,a}\big) - \rho t\big(i,a,e^k_{i,a}\big) \\ & + Q^k\big(e^k_{i,a}, \pi\big(e^k_{i,a}\big)\big) - Q^k(i,a)\big], \end{aligned} \tag{12}$$

where π is the policy being evaluated and where the term $e^k_{i,a}$ is a random variable that denotes the state to which the Markov chain jumps when action a is taken in the kth decision-making epoch and the system state is i. In this context, we can also define a transformation F on the vector Q^k as follows:

$$F(Q^k)(i,a) = \sum_j p(i,a,j)[r(i,a,j) - \rho t(i,a,j) + Q^k(j,\pi(j))]. \tag{13}$$

It can be shown that the scheme in (12) is of the form given in Theorem 2. The details of this are relegated to Appendix A.2.

Then all we need to do is to show that the conditions of Theorem 2 are satisfied. The condition of Lipschitz continuity (Condition 1) can be shown from the fact that the map f is linear everywhere in Q. Condition 2 can be shown by a learning rate (β) such as $1/k$. Condition 3 stems from the fact that the noise term w is the difference between a random value and its mean. Its variance can be easily bounded by a function of the square of the iteration. (Also its variance is bounded since the iterate Q itself is bounded, which is Condition 4.)

To prove boundedness (Condition 4) of the Q-factor, one has to show that the transformation (13) has a contraction property. Under assumption A, it is not hard to show that this is the case using a result from Bertsekas and Tsitsiklis (1996) and Littman (1996) (see Appendix A.1 for details). Then it can be shown that F is a pseudo-contraction (see p. 155 of Bertsekas & Tsitsiklis, 1996) since it is a contraction. Using the property of pseudo-contraction, it is not difficult to prove boundedness of the iterate via a standard result from Tsitsiklis (1994).

Since the transformation F is a contraction under Assumption A of Theorem 3, the differential equation $\frac{dQ(t)}{dt} = F(Q(t)) - Q(t)$ has a unique asymptotically stable equilibrium point, which is the fixed point of $F(Q)$. This proves that Condition 5 is satisfied.

Thus we have shown using Theorem 2 that the algorithm actually converges to the fixed point of $F(Q)$. It remains to be shown that the algorithm makes the Q-factors converge to the Q-factors associated with the policy. Now since the transformation $F(Q)$ is the Bellman transformation associated with the policy (see Step 2 of policy iteration details of Section 3.1), its fixed point is indeed the vector of Q-factors associated with the policy being evaluated. $\square$

Finite truncation of the policy evaluation phase. Since Step 3 must terminate in a finite number of iterations, the estimates of the Q-factors obtained will not equal their limiting values. It is to be understood that if the estimated Q-factors are far from their actual values, then the policy selected in the next visit to Step 2 may be different from that selected with the limiting values of the Q-factors. However, with a finite termination of the policy evaluation phase, we shall never obtain the "limiting" estimates. The fact that limiting values exist arises from Theorem 2, which states that the sequence converges.

Let $Q_1^*(i)$ and $Q_2^*(i)$ denote respectively the highest and the second highest *limiting* values of the Q-factors for state $i \in S$. Let $\tilde{Q}_1^k(i)$ and $\tilde{Q}_2^k(i)$ denote the same for the estimates in the kth iteration. Further, let $e_l^k(i)$ denote the absolute value of the difference in the actual value and the estimate in the kth iteration of $Q_l^*(i)$, where $l = 1, 2$ and $i \in S$. That is, for $l = 1, 2$,

$$e_l^k(i) = \left|\tilde{Q}_l^k(i) - Q_l^*(i)\right|.$$

We, then, have four cases. For each $i \in S$ and any finite k

$$\begin{array}{llll}
\textit{Case 1.} & \tilde{Q}_1^k(i) = Q_1^*(i) - e_1^k(i), & \text{and} & \tilde{Q}_2^k = Q_2^*(i) + e_2^k(i), \\
\textit{Case 2.} & \tilde{Q}_1^k(i) = Q_1^*(i) - e_1^k(i), & \text{and} & \tilde{Q}_2^k = Q_2^*(i) - e_2^k(i), \\
\textit{Case 3.} & \tilde{Q}_1^k(i) = Q_1^*(i) + e_1^k(i), & \text{and} & \tilde{Q}_2^k = Q_2^*(i) + e_2^k(i), \quad \text{and} \\
\textit{Case 4.} & \tilde{Q}_1^k(i) = Q_1^*(i) + e_1^k(i), & \text{and} & \tilde{Q}_2^k = Q_2^*(i) - e_2^k(i).
\end{array}$$

Let $D(i) = Q_1^*(i) - Q_2^*(i)$ for all $i \in S$.

We now claim that for a given state i, in the kth iteration, if each of $e_1^k(i)$ and $e_2^k(i)$ is less than $D(i)/2$, then $\tilde{Q}_1^k(i) > \tilde{Q}_2^k(i)$ for all i. This will imply that if the policy evaluation is terminated after k iterations, the correct policy will be selected in Steps 4 and 2.

Let us consider Case (1). What we are about to show can be shown for each of the other Cases. For any $i \in S$ and any finite k,

$$\begin{aligned}
\tilde{Q}_1^k(i) - \tilde{Q}_2^k(i) &= Q_1^*(i) - Q_2^*(i) - e_2^k(i) - e_1^k(i) \text{ from Case (1)} \\
&= D(i) - \left(e_1^k(i) + e_2^k(i)\right) > D(i) - D(i) = 0.
\end{aligned}$$

This implies that for every $i \in S$ and any finite value of k

$$\tilde{Q}_1^k(i) > \tilde{Q}_2^k(i),$$

which proves our claim.

Now, from Theorem 2, it follows that for each $i \in S$,

$$\lim_{k\to\infty} \tilde{Q}_1^k(i) = Q_1^*(i), \quad \text{and} \quad \lim_{k\to\infty} \tilde{Q}_2^k(i) = Q_2^*(i).$$

Hence, for a given $\epsilon > 0$, there exists a k_1 such that for all $k \geq k_1$

$$\left|\tilde{Q}_1^k(i) - Q_1^*(i)\right| < \epsilon.$$

Similarly, for a given $\epsilon > 0$, there exists a k_2 such that for all $k \geq k_2$

$$\left|\tilde{Q}_2^k(i) - Q_2^*(i)\right| < \epsilon.$$

Now, selecting $\epsilon = D(i)/2$ (where $D(i)$ must be greater than 0; note that when $D(i) = 0$, both actions are equally good and the problem is trivial, and by its definition $D(i)$ cannot be less than 0) and $K \equiv \max\{k_1, k_2\}$, we have that for every $i \in S$ and all $k \geq K$,

$$\left|\tilde{Q}_1^K(i) - Q_1^*(i)\right| < \epsilon = D(i)/2,$$

that is, for every $i \in S$ and all $k \geq K$,

$$e_1^k(i) < D(i)/2.$$

Similarly, it follows by using the same ϵ that, for every $i \in S$ and all $k \geq K$,

$$e_2^k(i) < D(i)/2.$$

Then from the claim above, it follows that the correct policy will be selected in the policy improvement stage with a number of iterations equaling or exceeding K. But K is a finite number. (The analysis presented above can be extended easily to three or more actions.) Thus we are done with ensuring that finite termination of policy evaluation with a suitable number of iterations leads to the correct policy being selected in the policy improvement step.

Incomplete evaluation of ρ. Step 2 of the algorithm (see figure 1) involves the calculation of the *estimate* of the average reward of a given policy via simulation. The estimate is Inexact, and it remains to be shown that the error introduced in the Q-factors due to this can be made arbitrarily small. If this can be shown, our proof of Theorem 3 will be complete.

Let us denote the actual average reward of a policy by ρ^* and the estimate with k replications by ρ^k. Let $Q^k(i, a)$ denote the Q-factor associated with state-action pair (i, a) in the kth iteration of policy evaluation of the algorithm. Let $\bar{Q}^k(i, a)$ denote the same factor, when the actual value of ρ is used. Let us define $O(\beta^2)$ as a function of the order of β^2 or higher powers of β. We now claim that:

$$|Q^k(i, a) - \bar{Q}^k(i, a)| \leq O(\beta^2) \quad \text{for all } (i, a) \tag{14}$$

if $Q^0(i, a) = \bar{Q}^0(i, a)$ for all values of (i, a). This will ensure that by selecting a suitable small value for β, the difference between the Q-factor that uses an estimate of ρ and the one that uses the actual value of ρ can be made arbitrarily small. We shall use an induction argument to prove our claim.

From Step 3c of the algorithm it follows that:

$$Q^{k+1}(i, a) = (1 - \beta)Q^k(i, a) + \beta[r(i, a, j) - \rho t(i, a, j) + Q^k(j, b)]. \tag{15}$$

where b depends on the policy being evaluated. If the actual value of ρ, which we denote by ρ^a, is used, the transformation in Step 3c can be written as:

$$\bar{Q}^{k+1}(i,a) = (1-\beta)\bar{Q}^k(i,a) + \beta[r(i,a,j) - \rho^a t(i,a,j) + \bar{Q}^k(j,b)] \tag{16}$$

It follows from the above and from $Q^0(i,a) = \bar{Q}^0(i,a)$ for all values of (i,a) that

$$Q^1(i,a) - \bar{Q}^1(i,a) = \beta[\rho - \rho^a]t(i,a,j),$$

which implies that:

$$|Q^1(i,a) - \bar{Q}^1(i,a)| = \beta|\rho - \rho^a|t(i,a,j). \tag{17}$$

In our algorithm, ρ is estimated by sampling (independent replications in a simulator) and the strong law of large numbers (see Law & Kelton, 1999) applies. From the strong law of large numbers, it follows that

$$|\rho - \rho^a|$$

can be made arbitrarily small. Therefore, for a given value of $\beta > 0$, there exists a number of samples (replications), such that:

$$|\rho - \rho^a| < \beta. \tag{18}$$

From the above, line (17) can be written as:

$$|Q^1(i,a) - \bar{Q}^1(i,a)| = \beta|\rho - \rho^a|t(i,a,j) < \beta\beta t(i,a,j) = O(\beta^2),$$

which proves our claim in inequality (14) when $k = 0$. Now, assuming that the claim is true when $k = n$, we have that:

$$|Q^n(i,a) - \bar{Q}^n(i,a)| \leq O(\beta^2). \tag{19}$$

From transformations (15) and (16), we can write that:

$$\begin{aligned} Q^{n+1}(i,a) - \bar{Q}^{n+1}(i,a) = (1-\beta)[Q^n(i,a) - \bar{Q}^n(i,a)] + \beta[\rho - \rho^a]t(i,a,j) \\ + \beta[Q^n(j,b) - \bar{Q}^n(j,b)]. \end{aligned}$$

From the above, one has that:

$$\begin{aligned} |Q^{n+1}(i,a) - \bar{Q}^{n+1}(i,a)| &\leq (1-\beta)|[Q^n(i,a) - \bar{Q}^n(i,a)]| + \beta|[\rho - \rho^a]|t(i,a,j) \\ &\quad + \beta|[Q^n(j,b) - \bar{Q}^n(j,b)]| \\ &< |[Q^n(i,a) - \bar{Q}^n(i,a)]| + \beta|[\rho - \rho^a]|t(i,a,j) \\ &\quad + |[Q^n(j,b) - \bar{Q}^n(j,b)]| \\ &\leq O(\beta^2) + \beta\beta t(i,a,j) + O(\beta^2) \\ &= O(\beta^2). \end{aligned} \tag{20}$$

Line (21) follows from inequalities (19) and (18). From the above, our claim is proved for $k = n + 1$, which implies that the claim is true for any k. Since β is a small quantity, the square of β is even smaller. By choosing a suitable value for β, the error in the Q-factors (due to the use of an estimate of ρ) can be made arbitrarily small. The proof of Theorem 3 is thus complete.

7. Conclusions

In this paper, an asynchronous RL algorithm, which is based on policy iteration, for solving MDPs and SMDPs for long-run average reward was presented. An algorithm based directly on policy iteration for the average reward case was missing in the literature. The algorithm is especially useful in SMDPs, where the value iteration procedure is approximate. It may be concluded that the algorithm performed satisfactorily on a large-scale problem from the real world and was able to clearly outperform a widely used heuristic in that area. Theoretical convergence of the algorithm was also shown via the rigorous ODE method. Extending this algorithm to partially observable Markov decision problems (POMDPs), and testing this algorithm on other industrial problems are interesting topics for further research.

A. Appendices

A.1. Contraction Property of transformation (13)

Theorem 4. *Consider the transformation* (13) *under assumption A of Theorem* 3. *There exists a vector ξ such that mapping F is a contraction mapping with respected to a weighted sup-norm $\|.\|_\xi$.*

Comment: It must be emphasized that the result we are about to present holds for the SMDP. A shortest stochastic path problem is invoked only for the sake of establishing a relation for the transition probabilities of our SMDP—a relation that can be used to prove the result. The proof is due to Littman (1996) and was also independently discovered by Bertsekas and Tsitsiklis (1996) (p. 23, Proposition 2.2). Even though, in the literature, the proof has been quoted in the shortest stochastic path context, it can be and has been used elsewhere because it pertains to the mapping. Any average reward problem under assumption A of Theorem 3, can be viewed as a stochastic shortest path problem (SSPP) by viewing successive visits to any recurrent state in the average reward problem (see Assumption A of Theorem 3) as cycles, each of which can be considered as a trajectory of an SSPP; the termination state of the latter is a recurrent state in the average reward problem. The transition probabilities do not change in their values. This result is related to the transition probabilities and hence will apply to any problem with the same transition probabilities.

Proof: We first consider a *new, fictitious* shortest stochastic path problem, whose transition probabilities are identical to the SMDP under consideration (as discussed above this is always possible under assumption A), but all the immediate transition costs are -1 and the immediate transition times are 1. $\square$

Then we have that for all $i = 1, 2, \ldots, n$ and stationary policies μ,

$$D(i,a) = -1 + \min_{v \in U(j)} \sum_j p(i,a,j)D(j,v) \leq -1 + \sum_j p(i,\mu(i),j)D(j,\mu(j)), \tag{21}$$

where $D(i, a)$ is the Q-factor for the state-action pair (i, a). Now setting μ to π, the policy being evaluated in map (13), we can write (21) as:

$$D(i,a) \leq -1 + \sum_j p(i,\pi(i),j)D(j,\pi(j)). \tag{22}$$

We shall make our notation a little more compact by replacing (i, a) by k and $(j, \pi(j))$ by l, where k and l denote the state-action pairs and take values in $\{1, 2, 3, \ldots, N\}$. Thus $D(i, a)$ will be replaced by $D(k)$. Let us now define a vector ξ as follows for all $k \in \{1, 2, 3, \ldots, N\}$:

$$\xi(k) = -D(k).$$

Then for all k, we have from the definition of $D(k)$ above (Eq. (22)), $D\,(k) \leq -1$, which implies that $\xi(k) \geq 1$ for $k = 1, 2, 3, \ldots, N$. For all stationary policies μ, we then have from (22) that for all k,

$$\sum_j p(i,\pi(i),j)\xi(l) \leq \xi(k) - 1 \leq \zeta\xi(k), \tag{23}$$

where ζ is defined by:

$$\zeta = \max_{k=1,2,\ldots,N} \frac{\xi(k)-1}{\xi(k)} < 1, \tag{24}$$

where the last inequality follows from the fact that $\xi(k) \geq 1$. Thus we have:

$$\sum_j p(i,\pi(i),j)\xi(l) \leq \zeta\xi(k), \tag{25}$$

which is true of the transition probabilities of the SMDP. We now turn our attention to the SMDP in question. From the definition of transformation F in Eq. (13), we can write:

$$FQ_1(i,a) - FQ_2(i,a) = \sum_j p(i,\pi(i),j)[Q_1(j,\pi(j)) - Q_2(j,\pi(j))]. \tag{26}$$

Using our compact notation, we can write the above as:

$$FQ_1(k) - FQ_2(k) = \sum_j p(i,\pi(i),j)[Q_1(l) - Q_2(l)]. \tag{27}$$

Then we have that:

$$\begin{aligned}
|FQ_1(k) - FQ_2(k)| &\le \sum_j p(i, \pi(i), j)|Q_1(l) - Q_2(l)| \\
&\le \sum_j p(i, \pi(i), j) \max_l |Q_1(l) - Q_2(l)| \\
&= \sum_j p(i, \pi(i), j)\xi(l)\|Q_1 - Q_2\|_\xi \\
&\le \zeta\xi(k)\|Q_1 - Q_2\|_\xi,
\end{aligned}$$

where we define the weighted max norm of $X(k)$ with respect to ξ as:

$$\|X\|_\xi = \max_k \frac{|X(k)|}{\xi(k)}.$$

From the above, we can write:

$$\frac{|FQ_1(k) - FQ_2(k)|}{\xi(k)} \le \zeta \|Q_1 - Q_2\|_\xi.$$

Since the above holds for all values of k, it also holds for the value of k for which its left hand side is maximized. In other words, we have that:

$$\|FQ_1 - FQ_2\|_\xi \le \zeta \|Q_1 - Q_2\|_\xi,$$

which proves the result.

A.2. A part of the proof of Theorem 3

We first need to define the functions f and w, which appear in Theorem 2, in terms of our algorithm.

Let us define a transformation H on the same vector as follows:

$$H(Q^k)(i,a) = \left[r\left(i, a, e^k_{i,a}\right) - \rho t\left(i, a, e^k_{i,a}\right) + Q^k\left(e^k_{i,a}, \pi\left(e^k_{i,a}\right)\right)\right],$$

Now if we define the noise term w^k as:

$$w^k = H(Q^k) - F(Q^k),$$

and if we define a function f as:

$$f(Q^k) = F(Q^k) - Q^k,$$

then we can write the updating transformation in our algorithm (Eq. (12)) as:

$$Q^{k+1}(i, a) = Q^k(i, a) + \beta[f(Q^k) + w^k],$$

which is of the same form as the updating scheme defined for Theorem 2.

Acknowledgments

The author would like to thank the Action Editor—Professor Andrew Barto—and the three anonymous reviewers for their comments. The author also thanks Jane M. Fraser for her help in reading the manuscript and making helpful suggestions. Finally, the author acknowledges support from the National Science Foundation Grant No: DMI:0114007.

References

Abounadi, J., Bertsekas. D., & Borkar, V. (1998). Learning algorithms for Markov decision processes with average cost. Technical Report, LIDS-P-2434, MIT, MA.

Belobaba, P. (1989). Application of a probabilistic decision model to airline seat inventory control. *Operations Research*, *37*, 183–197.

Bertsekas, D. (1995a). *Dynamic programming and optimal control*. Belmont, MA: Athena Scientific.

Bertsekas, D. (1995b). A new value iteration method for the average cost dynamic programming problem. Technical Report LIDS-P-2307, MIT, MA.

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Borkar, V. (1998). Asynchronous stochastic approximation. *SIAM Journal of Control and Optimization, 36:3*, 840–851.

Bradtke, S., & Duff, M. (1995). *Reinforcement learning methods for continuous-time Markov decision problems*. In *Advances in Neural Information Processing Systems 7*. Cambridge, MA: MIT Press.

Brumelle, S., & McGill, J. (1993). *Airline Seat Allocation With Multiple Nested Fare Classes Operations Research, 41*, 127–137.

Chatwin, R. (1998). Multiperiod Airline Overbooking With A Single Fare Class. *Operations Research, 46:6*, 805–819.

Curry, R. (1990). Optimal airline seat allocation with fare classes nested by origins and desitnations. *Transportation Science, 24*, 193–204.

Das, T., Gosavi, A., Mahadevan, S., & Marchalleck, N. (1999). Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science, 45:3*, 560–574.

Davis, P. (1994). Airline ties profitability yield to management. *SIAM News, 27:5*.

Glover, F., Glover, R., Lorenzo, J., & McMillan, C. (1982). The passenger-mix problem in the scheduled airlines. *Interfaces, 12*, 73–80.

Gosavi, A. (1999). An Algorithm for Solving semi-Markov Decision Problems Using Reinforcement Learning: Convergence Analysis and Numerical Results. Unpublished Ph.D. Dissertation, Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL.

Gosavi, A. (2003). A Reinforcement Learning Algorithm for Solving Markov and Semi-Markov Decision Problems Under Long-Run Average Cost. *European Journal of Operational Research*, to appear.

Gosavi, A., Bandla, N., & Das, T. (2002). Airline seat allocation among multiple fare classes with overbooking. *IIE Transactions, 34:9*, 729–742.

Horner, P. (2000). Mother, father of invention produce golden child: Revenue management— A Sabre story. *ORMS Today, 27:3*, 75–77.

Howard, R. (1971). *Dynamic probabilistic systems Volume II: Semi-Markov decision processes*. New York, NY: John Wiley and Sons.

Konda, V., & Borkar, V. (1999). Actor-critic type learning algorithms for Markov decision processes. *SIAM Journal on Control and Optimization, 38:1*, 94–123.

Kushner, H., & Clark, D. (1978) *Stochastic approximation methods for constrained and unconstrained systems.* New York, NY: Springer Verlag.

Law, A., & Kelton, W. (1999). *Simulation modeling and analysis, 3rd edition*. New York, NY: McGraw Hill.

Littlewood, K. (1972). Forecasting and control of passenger bookings. In *Proceedings of the 12th AGIFORS (Airline Group of the International Federation of Operational Research Societies) Symposium* (pp. 95–117).

Littman, M. (1996). Algorithms for sequential decision-making. Unpublished Ph.D. Thesis, Brown University, Providence, R.I.

Mahadevan, S. (1994). To discount or not to discount: A case study comparing R-learning and Q-learning. In *Proceedings of the 11th International Conference on Machine Learning* (pp. 164–172). New Brunswick, NJ.

Mahadevan, S. (1996a). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning, 22:1*, 159–195.

Mahadevan, S. (1996b). An average-reward reinforcement learning algorithm for learning bias-optimal policies. In *Proceedings of the 13th National Conference on Artificial Intelligence* (pp. 875–880). Cambridge, MA: MIT Press.

McGill, J., & van Ryzin, G. (1999). Revenue management: Research overview and prospects. *Transporation Science, 33:2*, 233–256.

Puterman, M. (1994). *Markov decision processes*. New York, NY: Wiley Interscience.

Robinson, L. (1995). Optimal and approximate control policies for airline booking with sequential nonmonotonic fare classes. *Operations Research, 43*, 252–263.

Rummery, G., & Niranjan, M. (1994). On-line Q-Learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University, England.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth Annual Conference on Machine Learning* (pp. 298–305). Morgan Kaufmann.

Shlifer, E., & Vardi, Y. (1975). An airline overbooking policy. *Transportation Science, 9*, 101–114.

Subramaniam, J. Stidham, S., & Lautenbacher, C. (1999). Airline yield management with overbooking, cancellations and no-shows. *Transportation Science, 33:2*, 147–167.

Sutton, R. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8* (pp. 1038–1044). Cambridge, MA: MIT Press.

Sutton, R.S., & Barto, A. (1996). *Reinforcement learning*. Cambridge, MA: MIT Press.

Tsitsiklis, J. (1994). Asynchronous stochastic approximation and Q-Learning. *Machine Learning, 16*, 185–202.

van Ryzin, G., & McGill, J. (2000). Revenue management without forecasting or optimization: An adaptive algorithm for determining seat protection levels. *Management Science, 46*, 568–573.

Wollmer, R. (1992). An airline seat management model for a single leg route when lower fare classes book first. *Operations Research, 40*, 26–37.

Received August 2, 2000
Revised January 15, 2003
Accepted January 15, 2003
Final manuscript March 11, 2003