A Reinforcement Learning Approach to Airline Seat Allocation for Multiple Fare Classes with Overbooking

Abhijit Gosavi
(gosavi@uscolo.edu)
College of Engineering, Industrial Engineering Program
University of Southern Colorado, Pueblo, CO-81001
Naveen Bandla
(Naveen_Bandla@sabre.com)
Sabre Technologies Inc, Dallas, Texas
Tapas K. Das
(das@eng.usf.edu)
Department of Industrial and Management Systems Engineering
University of South Florida, Tampa, FL 33620

June 11, 2000

Abstract

The airline industry strives to maximize the revenue obtained from the sale of tickets on every flight. This is referred to as *yield management* and it forms a crucial aspect of airline logistics. Ticket pricing, seat or discount allocation, and overbooking are some of the important aspects of a yield management problem. Though ticket pricing is usually heavily influenced by factors beyond the control of an airline company, significant amount of control can be exercised over the seat allocation and the overbooking aspects. A realistic model should consider *multiple fare classes*, *overbooking* of the flight, *concurrent* demand arrivals of passengers from the different fare classes, and *class-dependent, random cancellations*. Accommodating all these factors in one optimization model is a challenging task because that makes it a very large-scale stochastic optimization problem. Almost all papers in the existing literature accommodate only a subset of these factors in order to make the model tractable. We consider all these factors and cast the problem as a Semi-Markov decision problem (SMDP) under the average reward optimizing criterion over an infinite time horizon. We solve it using a stochastic optimization technique called Reinforcement Learning (RL). Not only is RL able to scale up to a huge state-space but because it is simulation-based it can also

handle complex modeling assumptions such as the ones mentioned above. The state space of the problem scenarios considered here is non-denumerable; its *countable* part being of the order of 10^9. Our solution procedure involves a multi-step extension of the SMART algorithm which is based on the one-step Bellman equation. Numerical results presented here show that our approach is able to outperform a heuristic, namely the EMSR heuristic, which is *widely used in the airline industry.* We also present a detailed study of the sensitivity of some modeling parameters via a full factorial experiment.

1 Introduction

Since the deregulation of the airline industry in 1978, airlines have been allowed to choose their own market segments, decide their own routes, and set their own fares as long as they comply with the safety and security regulations enforced by the Federal Aviation Administration (FAA). The fierce competition that has ensued in the airline industry as a result of this has forced every airline into making efforts at gaining a competitive edge in the market. As a result, airlines are turning to advanced optimization techniques to develop decision support systems for management and control of airline operations. An important goal of the airline industry 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 [11]: "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."

Ticket pricing, seat or discount allocation, and overbooking are some of the important aspects of the yield management problem. Though ticket pricing is usually heavily influenced by external factors (such as prices set by the competitors), significant control can be exercised by airlines over the seat allocation and the overbooking aspects. We examine 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 (stochastic dynamic programming) 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 takes all these factors into account and generates optimal or near-optimal results. According to a recent review paper on revenue management written by McGill and Van Ryzin [19]: "Recently developed approximation methods for dynamic programming and stochastic programming may be useful in revenue management. Good references for these approximation methods are the books by Bertsekas and Tsitsiklis [4]..." Our paper makes an attempt in this direction. We account for all the factors mentioned above and use an approximate dynamic programming approach to solving this problem.

Henceforth, in this paper, we will refer to the problem of seat allocation and overbooking as the yield management problem. This problem was identified as the hottest area of new research in traffic management by Chatwin [7]. Robinson [22] stressed on the need for considering concurrent arrivals of fare classes, since almost all existing modeling approaches consider that the fare classes arrive sequentially (lower classes first and so on).

The seat allocation problem arises from the airlines' practice of selling seats within the same cabin of a flight at different prices to the customers of different classes. The airlines protect some seats from the lower revenue fare classes in order to be able to satisfy demands from the higher revenue classes; the obvious question here is what is the optimal level of protection. The overbooking problem arises from the fact that customers do cancel their tickets and often fail to show up at the flight time (no-shows). The airlines tend to overbook in anticipation of such cancellations and no-shows to avoid flying with empty seats. (A seat in an airplane is a perishable inventory the value of which vanishes as soon as the gate closes for a flight. Airline seat allocation problem has been studied in the context of inventory control by Topkis [32].)

However, overbooking has its downside in that airlines run the risk of not having enough seats for all the ticket-holders. When such a situation arises, airlines are forced to deny (bump) boarding request to the extra ticket- holders and pay a penalty in two ways: directly in the form of financial compensations to the bumped passengers, and indirectly through loss of customer goodwill. Hence, a prudent choice of overbooking policy that maximizes the yield is called for. Smith et al. [27] estimated that effective yield management can save airlines hundreds of millions of dollars each year.

Passenger classification 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. Airlines usually use a combination of factors in determining the fare class of the passenger. Also different airline companies use different factors in classification. We briefly discuss, next, the different factors used in such a classification to motivate the assumption of multiple fare classes in airline yield management literature.

An important factor used in classification is the time of the request for reservation. Passengers who book in advance are thrown into lower fare-classes and those who come later have to pay higher fares. If classification is done solely on the basis of this factor, it is alright 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.

Another important factor that is used in classification is the itinerary of the passenger. To see how an origin-and-destination based (itinerary-based) passenger classification works, consider a single flight leg from Detroit to New York. The flight in addition to carrying passengers whose origin is Detroit and destination New York, is likely to be carrying passengers from other longer itineraries, such as Tampa-Atlanta-Detroit-New York, Dallas-Detroit-New York, and Phoenix-Atlanta-Detroit-New York. If the passengers whose itinerary originates from Detroit form the highest class, those flying from Dallas (or Atlanta) to New York via

Detroit form the middle class, and those flying from Tampa (or Phoenix) to New York via Atlanta and Detroit (and are on the third leg of their itinerary) form the lowest fare class of passengers in the plane. See also Figure 1, which shows how the class of the passenger can depend on the itinerary. A circle, in Figure 1, 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 factor happens to be one of the factors used to classify passengers, sequential passenger arrival will be a poor assumption and as suggested by Robinson [22], a concurrent arrival pattern must be assumed.

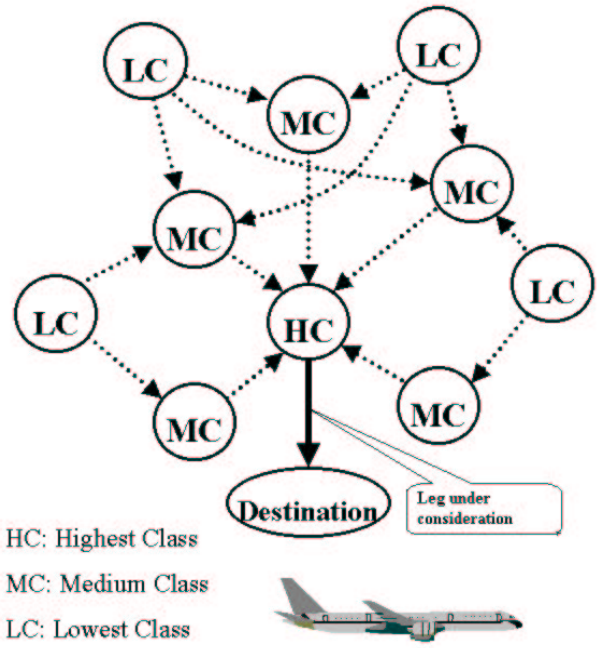

Figure 1: A schematic showing classification of customers based on the origin (circle) and the destination, in one particular leg of an airline flight

A recent paper by Chatwin [7] examines the optimality criteria for a multiperiod airline overbooking problem, and then applies the model to the seat allocation problem. The process of maximizing yield subject to constraints, such as the limitations on the number of fare

classes available and the number of seats available in each class was studied comprehensively by Belobaba [3]. This work uses a generalized (multiperiod) version of Littlewood's [16] equation to obtain allocations for more than two fare classes. Belobaba's method, named EMSR (Expected Marginal Seat Revenue) model, results in a fixed quantity of seats allocated to each fare class. Brumelle and McGill [6] studied the same problem. They assumed that seats are booked in a so-called "nested" fashion and that lower fare classes book before higher ones. Brumelle and McGill [6], Curry [8], and Wollmer [34] independently reported the non-optimality of the EMSR model developed by Belobaba for all fare classes. Van Ryzin and McGill [24] considered the airline yield management problem without cancellations and no-shows and they assumed that lower fare classes book strictly before higher ones. They proposed a simple adaptive approach which solves the Littlewood's equation using a Robbins-Monro stochastic approximation scheme. Glover et al. [12] and Shlifer and Vardi [26] are some of the other researchers who have done significant work in this area. Howard [14] set up the airline overbooking problem for a single fare class as a Markov decision problem in which the system state at any time is given by the number of reservations and the time remaining to the departure of the flight. Howard proposed the use of the value iteration method to obtain the optimal policy for the problem of overbooking. However, only very small problems can be solved with this approach because of the computational limitations of value iteration. Another paper that models this problem as an MDP is Subramaniam *et al* [28]. They discretize the time horizon and use 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 nor do we discretize the time. We retain it as an SMDP and consider all the complexities mentioned above. We then use a reinforcement learning method to solve this SMDP. The reader is referred to two excellent survey papers - one by Kaebling *et al* [15] for an overview of RL methods in general and the other by Mahadevan [17] for RL methods related to average reward.

To our knowledge this is the first paper that uses reinforcement learning as a method for finding an optimal solution to the yield management problem. Our paper considers a model for multiple fare classes assuming all the complicating modeling assumptions such as random cancellations, overbooking and concurrent demand arrivals thereby allowing classification based on a combination of factors. We have shown that when all these assumptions are made, the EMSR model, which is widely used in the industry [11], can be outperformed using our reinforcement learning model. Some of the work presented here has appeared in the Master's thesis of Bandla [1]. The rest of this paper is organized as follows. In Section 2, we give a formal description of the problem and the semi-Markov model. In Section 3, we discuss the methods that can be used to solve an SMDP. In Section 4, we present a new RL algorithm named λ-SMART, that we have used in solving the underlying SMDP of the yield management problem. In Section 5, we describe the sample problems we have used for testing the efficacy of λ-SMART. We also discuss how this algorithm is implemented on the problem using a neural network. Section 5 also contains an account of the EMSR model. In Section 6, we present results from the sample problems studied and some discussion on the results obtained. Section 7 shows the results of a sensitivity analysis carried out on some of the key modeling parameters. Section 8 contains some concluding remarks and an account of the possible future work in this area.

2 Problem Description

The flight considered in this paper is a *single leg* of a complete origin-and-destination itinerary. It is assumed that there are multiple types of customers who request reservations for the seats available in the coach cabin of the aircraft. Whenever a customer requests a reservation, the airline needs to make a decision whether to accept or deny the reservation. The requests for tickets of different classes are assumed to arrive according to independent Poisson processes. Passengers of different classes cancel their reservations with fixed proba-

bilities. (Hence, the number of cancellations in a period follow the binomial probability law. Chatwin [7] cites two studies that have confirmed the appropriateness of this assumption through case studies: Martinez and Sanchez [18] for Iberia Airlines, and Thompson [31] for Tasman Empire Airways.) The time of cancellation is uniformly distributed between the time of purchase and flight take-off. If a passenger with a confirmed ticket is not allowed to board the flight due to overbooking, the airline incurs a financial and goodwill cost (bumping cost). Our goal is to develop a strategy for seat allocation and overbooking so as to maximize the average reward earned by the airline per unit time. In the following section, we present an SMDP model for the problem.

2.1 SMDP Model

In order to model the yield-management problem as a semi-Markov decision problem (SMDP), we first define the system state-space. For n fare classes, the system state (ϕ) can be denoted by the vector

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

where c represents the class of the most recent customer (among n possible classes) seeking a ticket, $s_1, s_2, ..., s_n$ represent the number of seats sold in n classes, ψ_k, for $k\epsilon\{1, 2, ..., n\}$, 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 $n * M^n$, 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. Hence, the time epochs of customer request arrivals form the so-called *decision-making epochs*.

Let X_m and T_m denote the system state and time respectively at the mth decision-making epoch. We define two stochastic processes: $X = \{X_m : m \in \mathcal{N}\}$ and $T = \{T_m : m \in \mathcal{N}\}$ where $\mathcal{N}$ is the set of natural numbers. We also define a joint process $(X, T) = \{(X_m, T_m) : m \in \mathcal{N}\}$. It can be easily shown that

$$P[X_{m+1} = j, T_{m+1} - T_m <= t | X_0, ..., X_m; T_0, ..., T_m] = P[X_{m+1} = j, T_{m+1} - T_m <= t | X_m, T_m], \quad (1)$$

which means that the process (X, T) is a Markov renewal process. Then the decision process associated with (X, T) is a semi-Markov decision process, and X is a Markov chain underlying the Markov renewal process.

The process that tracks every state change (including customer request, cancellation, and flight departure) is referred to as the *natural process*, and the process embedded at the decision-making epochs (customer request arrivals) is referred to as the *decision process*. Between two successive decision-making epochs, it is possible that several cancellations take place causing many changes in the system state, changes that are tracked only in the natural process. In the rest of this paper, a decision-making state will be referred to as state. 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.

Since our objective is to derive an optimal policy for the general yield management problem, it is necessary to define a performance metric for optimization. The metric used in this paper is average reward which is defined next. We first define the notation that will be used. Let S and A denote the set of decision-making states and the set of actions respectively. Let A_i

denote the set of actions that may be taken in state $i \in \mathcal{S}$. Also let $r(i,a,j)$ (for $i \in S$ and $a \in A_i$) denote the immediate reward gained when in the decision-making state i action a is chosen and the next decision-making state encountered by the system is j. Let $t(i,a,j)$ denote the time spent in this transition from state i to state j as a result of adopting action a in state i. Then the average reward of an SMDP starting at state i and continuing with policy π for an infinite time period can be given (using the renewal reward theorem [20], [23]) as

$$\rho^{\pi}(i) = \frac{\lim_{N\to\infty} \frac{1}{N} E_i^{\pi} \left[\sum_{k=1}^{k=N} \left(r(i_k, a_k, i_{k+1})\right]\right.}{\lim_{N\to\infty} \frac{1}{N} E_i^{\pi} \left[\sum_{k=1}^{k=N} \left(t(i_k, a_k, i_{k+1})\right]\right.} \tag{2}$$

where i_k and a_k denote the state and action respectively at the kth decision-making epoch, and $E_i^{\pi}[.]$ denotes the expectation operator under policy π when the initial state is i. If the Markov chain X has a unichain transition probability matrix (see [20] for a detailed analysis and definition of a unichain transition probability matrix), the expected average cost does not vary with the initial state for any stationary policy.

The infinite horizon average reward SMDP with a recurrent class of states has a connection with the shortest stochastic path problem (SSPP) - a connection that will be used fruitfully in this paper. An SSPP contains a termination state which is an absorbing state, in which no reward is earned. The connection between the SSPP and average reward DP was first shown by Bertsekas [5]. The sequence of the states visited in an average reward problem over an infinite time horizon, which has a recurrent state s, can be divided into cycles marked by the successive visits to the recurrent state. Clearly each of these cycles starts and ends at the recurrent state s. Each of the cycles can also be viewed as an SSPP, whose termination state is s. Thus the average reward scenario can be viewed as a sequence of SSPPs. It must be emphasized here that the model used in this paper is not an SSPP but an SMDP - we only show the connection because the connection can be used to calculate the temporal difference in a convenient manner (explained in Section 3.2).

3 Solving SMDPs

In this section, we discuss how RL as a method has emerged from the dynamic programming (DP) framework. The theory of dynamic programming (see Puterman's textbook [20] for an in-depth discussion) forms a convergent framework that may be used to solve SMDPs.

3.1 Dynamic Programming

Dynamic programming was developed by Bellman [2] to solve MDPs and SMDPs. Bellman proved that the optimal policy for an SMDP may be obtained by iteratively solving a linear system of equations. The unknown in this linear system is the so-called *value function* - one for each state of the SMDP. There are several well known DP algorithms, such as policy iteration, value iteration, and linear programming, that find optimal solutions. But a drawback of these methods is that they require the calculation of the one-step transition probability matrices, the reward matrices, and the sojourn time matrices. The computation and storage of these matrices for problems with a very large state space can become almost impossible. Hence obtaining an optimal solution using DP algorithms is often quite difficult and for solving large-scale stochastic dynamic programs, one has to turn to other methods such as reinforcement learning and learning automata (Wheeler and Narendra [33]).

3.2 Reinforcement Learning

In recent years, an alternative approach called Reinforcement Learning (RL) has become a topic of intense research. It is a simulation-based method, rooted in the Bellman equation, which combines the principles of stochastic approximation (e.g. Robbins-Monro method [21]) and function approximation (e.g. neural networks and regression). Textbook treatment of this topic can be found in Sutton and Barto [29] and Bertsekas and Tsitsiklis [4]. Convergent algorithms based on this method have been shown to obtain near-optimal policies on

Markovian problems with a considerable reduction in computational effort.

Reinforcement learning has two distinct advantages over DP. The first advantage is that it can handle problems with complex transition mechanisms by making judicious use of the Robbins-Monro stochastic approximation algorithm thereby eliminating the need to compute or store the transition probabilities. Secondly, RL can integrate within it various function approximation methods (regression, neural networks, etc.), which can be used to approximate the value function even when the size of the state-space is gargantuan. Q-Learning is an RL algorithm based on value iteration and the Robbins-Monro stochastic approximation scheme. We next discuss some transformations related to it.

Q-Learning is a value iteration based RL method that solves the Bellman equation iteratively in an asynchronous style to obtain the optimal value function and the optimal policy. The strategy adopted in Q-Learning is to obtain the values of the so-called Q-factors - one for each state-action pair. The action that has the highest Q-factor for a state constitutes the optimal action for it. The optimal Q-factor - $Q(i,a)$ for a pair (i,a) with $i \in S$ and $a \in A_i$ is defined as follows,

$$Q(i,a) = \sum_{j \in S} p(i,a,j)[r(i,a,j) - \rho t(i,a,j) + R^*(j)], \tag{3}$$

where $r(i,a,j)$ and $t(i,a,j)$ are as defined before and ρ is the optimal average reward and $R^*(i)$ is the value function of dynamic programming for state i. The Bellman equation, which gives a recursive definition for the value function, in case of SMDPs [20] is given as follows:

$$R^*(i) = \max_{a \in A_i} \sum_{j \in S} p(i,a,j)[r(i,a,j) - \rho t(i,a,j) + R^*(j)]. \tag{4}$$

From equations (3) and (4), one has that

$$R^*(i) = \max_{a \in A_i} Q(i,a), \tag{5}$$

Using equation (5), equation (3) can be written as:

$$Q(i,a) = \sum_{j \in S} p(i,a,j)[r(i,a,j) - \rho t(i,a,j) + \max_{b \in A_j} Q(j,b)] \qquad \forall i. \tag{6}$$

Equation (6) forms the crucial transformation in any Q-learning algorithm. Its step-size version (also known as a *relaxed* version), is used more commonly in reinforcement learning. For a step-size γ where $\gamma \in (0,1]$, the step-size version is given as follows:

$$Q(i,a) = Q(i,a) + \gamma \sum_{j \in S} p(i,a,j)[r(i,a,j) - \rho t(i,a,j) + \max_{b \in A_j} Q(j,b) - Q(i,a)]. \tag{7}$$

The transformation in equation (7) may be viewed as a Robbins-Monro [21] stochastic approximation scheme, which is used to approximate a random variable with methods such as simulation. Use of the Robbins-Monro scheme makes it possible to replace the expectation (over j) by a single sample of the term within the square brackets in equation (7). Hence equation (7) may be written as:

$$Q(i,a) = Q(i,a) + \gamma[r(i,a,j) - \rho t(i,a,j) + \max_{b \in A_j} Q(j,b) - Q(i,a)] \qquad \forall (i,a). \tag{8}$$

The samples are generated through simulation. Transformation (8) forms the foundation for the new algorithm (λ-SMART) that we have developed and used to solve the yield management problem. The use of the step-size enables one to replace the expectation by the sample. We next discuss another concept which is central to the technique of RL.

Equation (8) can be rewritten as follows:

$$Q(i,a) = Q(i,a) + \gamma[TD], \tag{9}$$

where

$$TD = [r(i,a,j) - \rho t(i,a,j) + \max_{b \in A_j} Q(j,b) - Q(i,a)]. \tag{10}$$

The term TD is commonly referred to as the *temporal difference* for $Q(i,a)$. This concept is due to Sutton [30]. It can be used in estimating the value of a variable using simulation. The

term TD in equation (10) denotes the difference between the simulation estimate (explained below) of $Q(i,a)$ and its current estimate. Here is how. The simulation estimate of $Q(i,a)$ which is given by : $\{r(i,a,j) - \rho t(i,a,j) + \max_{b \in A_j} Q(j,b)\}$ is a function of the immediate reward in *one* transition of the SMDP and the current estimate is $Q(i,a)$ itself.

The temporal difference may also be calculated as a function of *all* the immediate rewards earned in the trajectory (see pg 387 of [4]). The algorithm presented in this paper seeks to replace TD as given in (10) by the sum of rewards earned in an infinitely long trajectory. The sum of rewards earned in an *infinitely* long trajectory may be approximated, in the case of the average reward problem, by the sum of the rewards earned in the *finite* number of states in a cycle of the underlying SSPP plus the value function of the recurrent state, which can be viewed as the termination state of the SSPP.

The temporal difference function is usually denoted by TD(λ) where λ is a weighing factor that weighs immediate rewards further in time by increasing powers of λ. The sum of the temporal differences in an infinite trajectory, denoted by TD(λ) is defined as :

$$TD(\lambda) = \sum_{m=k}^{\infty} \lambda^{m-k} d_m \tag{11}$$

where d_m denotes the temporal difference at the mth transition and λ takes a value between 0 and 1. (Here we assume that $0^0 = 1$.) For the average reward SMDP, the temporal differences do not have to be summed over an infinitely long trajectory as discussed in the preceding paragraph. Using this concept, we derive an algorithm that we have named $\lambda - SMART$, the details of which will be presented in the next Section. SMART is an acronym for Semi-Markov Average Reward Technique and the term λ stands for TD(λ). SMART is a one-step temporal difference algorithm (see Das *et al* [10]). λ-SMART sums the temporal differences over the entire trajectory as defined above. By varying the values of λ, a whole family of algorithms can be generated.

4 λ-SMART Algorithm

In what follows, we present the λ-SMART algorithm.

1. Define five sets that contain values obtained at the decision-making epochs in the trajectory. Let $\mathcal{A} = (i_0, i_1, ..., i_l, ...)$ denote the set of states visited in a trajectory, where i_l is the $(l+1)^{th}$ state visited in the trajectory, and $\mathcal{B} = (a_0, a_1, ..., a_l, ...)$ be the set of actions taken in those states. Also, let $\mathcal{C} = (r(i_0, a_0, i_1), r(i_1, a_1, i_2), ..., r(i_l, a_l, i_{l+1}), ...)$ and $\mathcal{D} = (\tau(i_0, a_0, i_1), \tau(i_1, a_1, i_2)..., \tau(i_l, a_l, i_{l+1}), ...)$ denote respectively the set of the immediate rewards earned and the set of sojourn times in the trajectory defined above. Also let $\mathcal{E} = (\alpha_0, \alpha_1, ...)$ and $\mathcal{F} = (g_0, g_1, ...g_l, ...)$ denote the set of the learning rates and the sequence of average reward values at the different states in the trajectory respectively. λ is a fixed number between 0 and 1. Set action values $Q(j, a) = 0$ for all $j \in S$ and $a \in A_j$. Set the cumulative reward $c_{new} = 0$ and the total time $t_{new} = 0$ and l the trajectory index to 0.

2. Start simulation in state s with decision-making epoch $m = 0$.

   While $m <$ MAX_STEPS do

   Let the system state at the m^{th} decision-making epoch be i_l where the subscript l indicates that it is the $(l+1)^{th}$ element in the current trajectory.

   (a) With probability $(1 - p_m)$, choose an action $a_l \in A_{i_l}$ for which $Q(i_l, a_l)$ is maximum, otherwise choose a random (exploratory) action from the set $\{A_{i_l} \setminus a_l\}$, where $p_m = f_1(m)$ (the function f_1 used to decay the probability of exploration).

   (b) Perform the chosen action. Let the system state at the next (i.e. $(m+1)^{th}$) decision epoch be i_{l+1}. Also let $\tau(i_l, a_l, i_{l+1})$ be the transition time, and $r(i_l, a_l, i_{l+1})$ be the immediate reward earned as a result of taking action a_l in state i_l.

   (c) If a nonexploratory action was chosen in step 2 (a)

- Set $c_{new} \leftarrow c_{new} + r(i_l, a_l, i_{l+1})$
- Set $t_{new} \leftarrow t_{new} + \tau(i_l, a_l, i_{l+1})$
- Set $g_{l+1} \leftarrow \frac{c_{new}}{t_{new}}$
- Set $\alpha_{l+1} \leftarrow f_2(m)$, where $f_2(.)$ is a function that is used to decay the learning rate.

Else,

Set $g_{l+1} \leftarrow g_l$.

(d) Add the $(l+2)^{th}$ elements in all the sets $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ and $\mathcal{F}$.

(e) If $i_{l+1} = s$, indicating that the end of the trajectory has been reached (from state s back to state s), then for $k = 0$ to l, set

$$Q(i_k, a_k) \longleftarrow Q(i_k, a_k) + \alpha_k \sum_{p=0}^{k} \lambda^p d_p \tag{12}$$

where the pth temporal difference d_p is given by

$$d_p = r(i_p, a_p, i_{p+1}) - g_p \tau(i_p, a_p, i_{p+1}) - Q(i_p, a_p) + Q(i_{p+1}, a_{p+1}) \tag{13}$$

Set $l = 0$ and reset all the sets to empty sets.

Else,

set $l \leftarrow (l+1)$.

(f) Set $m \leftarrow (m+1)$, and go to step 2 (a).

The learning rates $f_1(m)$ and $f_2(m)$ that we used are Darken-Chang-Moody (DCM) rates [9]. They work as follows: at the $(m+1)$th step, the rate x_{m+1} is given by $x_{m+1} = \frac{x_0}{1+u}$, where $u = \frac{m^2}{\tau+m}$. The values we used throughout our experiments are as follows: $x_0 = 0.005$ and $\tau = 10^{15}$ for the learning rate α, and $x_0 = 0.1$ and $\tau = 10^{15}$ for the exploration probability p.

5 Numerical Analysis

In this Section, we describe some of the problems selected for numerical analysis of the new algorithm λ-SMART. We also discuss some of its implementational issues. The section ends with a discussion on the heuristic used to benchmark the performance of λ-SMART.

5.1 Test problems

In order to evaluate the performance of λ-SMART we studied a set of sample seat allocation problems with the following characteristic parameters.

- The flight has three fare classes and the fare structure is represented by the vector $FS = (f_l, f_m, f_h, b)$, where f_l is the fare for the lowest class, f_m is the fare for the middle class, f_h is the fare for the highest class, and b is the bumping cost. Three different sets of fare structures (FS_j, $j = 1, 2, 3$) were considered.
- The customer arrival process is Poisson with parameter λ_a.
- The capacity of the aircraft is C.
- The customer distribution for class $i \epsilon \{l, m, h\}$ is represented by the vector $CD = (p_l, p_m, p_h)$, where p_l is the probability of a customer belonging to the lowest class, and p_m and p_h respectively for the middle and high classes. Clearly, $p_l + p_m + p_h = 1$. Like the fare structure, three different sets of customer distribution (CD_j, $j = 1, 2, 3$) were considered.
- 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.

The system state for a 3 fare class problem can be represented by the vector

$$\phi = (c, s_1, s_2, s_3, \psi_1, \psi_2, \psi_3, t).$$

We use 1 for the index of the lowest class, 2 for the middle class, and 3 for the highest class. Table 1 summarizes all the numerical problems that were examined using the λ-SMART algorithm. For problems 18, 19, and 20, cancellation probability is considered to be class dependent. For all the test problems, when booking for a new flight starts the system state is assumed to be the recurrent state s mentioned in Section 4. λ is set to 1. We found that for this problem the value of 1 for λ gave the best results. In the next Section, we describe the function approximation techniques used in the numerical study.

5.2 Function Approximation

For approximating the Q-values over the state-action space, we use a function approximation scheme by which a large number of Q-values are stored in the form of a small number of weights. A function approximation scheme is required because storing the Q-values for the huge state-action space is impossible. The state space is partitioned and encoded in a manner suitable for use with the neural network. After performing several experiments, we found that the system state is adequately represented by the following features : the class of the customer currently seeking a ticket, and the number of seats already sold in each of the classes.

This reduced state space was divided into subsets using hyperplanes and in each subset a linear *neuron* (explained below) was used to approximate the Q-value. This is equivalent to the approximation of a non-linear function by piecewise linear approximation. Each fare class forms a distinct subset, and within each fare class, subsets are formed by placing separating hyperplanes at 5 units. Thus for k classes and a flight capacity of M, there

would be $(kM/5)^+$ number of distinct subsets (for each action), where $(\beta)^+$ is the value of β rounded to the next higher integer.

A linear neuron is a two-layered neural network where the input layer contains the input nodes and the output layer contains the output node. The output node in our case is the Q-value and the input nodes denote the variables used to define the state-space. Our neurons have two input nodes each (see Figure 2). One node is the bias unit which is always turned on and has an input of 1. The other unit is the number of seats already sold in the class in which the reservation is sought. The input is normalized between 0 and 1. When, as a result of an action, a transition takes place to a new decision-making state, the Q-value for the action taken in the old decision-making state has to be updated. The net stores the Q-value for each state-action pair in form of the weights and hence the updating of the Q value is achieved by updating the weights that store this particular Q-value. The rule used to update the weights is the Widrow-Huff rule, also called the delta rule. The delta rule is implemented as follows. Whenever updating is required, the old Q value for the state-action pair in question is obtained from the network (explained below). The new Q value is then determined using the λ-SMART algorithm. The new Q value serves as the target value (t) in the scheme used to update the weights (explained below).

Consider a linear neuron with m input units and 1 output unit. The ith input is denoted by x_i and the weight connecting it to the output is denoted by w_i. Initialize all weights (w_i for all values of i) to small random numbers. Set max to some predetermined number and $iter = 1$ `Do until` $iter = max$

1. `Calculate` δ:

$$\delta = t - o \tag{14}$$

`where` t `is the target value of output unit (the new` Q `value) and` o `(the old`

Q value) is the network output of the output unit given by $o = \sum_{i=1}^{i=m} w_i x_i$.

2. Update each network weight w_i by the formula

$$w_i \leftarrow w_i + \Delta w_i \tag{15}$$

where $\Delta w_i = \eta \delta x_i$ and η is the learning rate.

3. Set $iter \leftarrow iter + 1$.

In the numerical problems that we studied, max was set to 2 and η was decayed according to the DCM scheme.

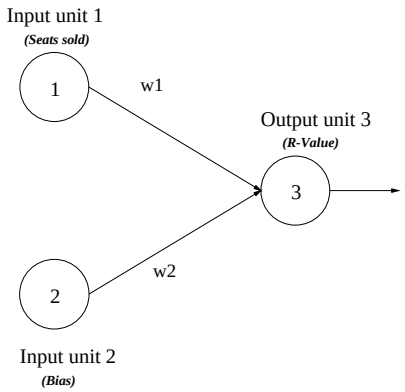

Figure 2: Schematic of a neuron used in function approximation

5.3 EMSR-Heuristic

In what follows, we present a heuristic algorithm to benchmark the performance of λ-SMART. Littlewood's equation has been used in the solution methodology developed by Belobaba [3] for the problem of assigning seats to different fare classes. We refer to the EMSR model of Belobaba as the EMSR (Expected Marginal Seat Revenue) heuristic. The

strategy adopted in the EMSR heuristic comprises 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 on any given day t, Littlewood's equation is solved to obtain the unknown quantities $S_j^i(t)$ for $i > j$, the equation being :

$$P(X_i(t) > S_j^i(t)) = f_j/f_i \qquad j = 1, ..., k, \tag{16}$$

where $X_i(t)$ denotes the number of requests for class i that will arrive in the remaining time till flight departure, $S_j^i(t)$ 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 $S_j^i(t)$ are obtained, the booking limit for a class j at time t may be obtained by:

$$BL_j(t) = C - \sum_{i>j} S_j^i(t) - \sum_{i>j} b_i(t), \tag{17}$$

where C is the capacity of the plane, and $b_i(t)$ is the number of reservations on class i till time t. 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 C^* is given by

$$C^* = C/(1-p).$$

The results for 20 numerical problem cases and the details of a sensitivity analysis performed on some key modeling parameters are presented next.

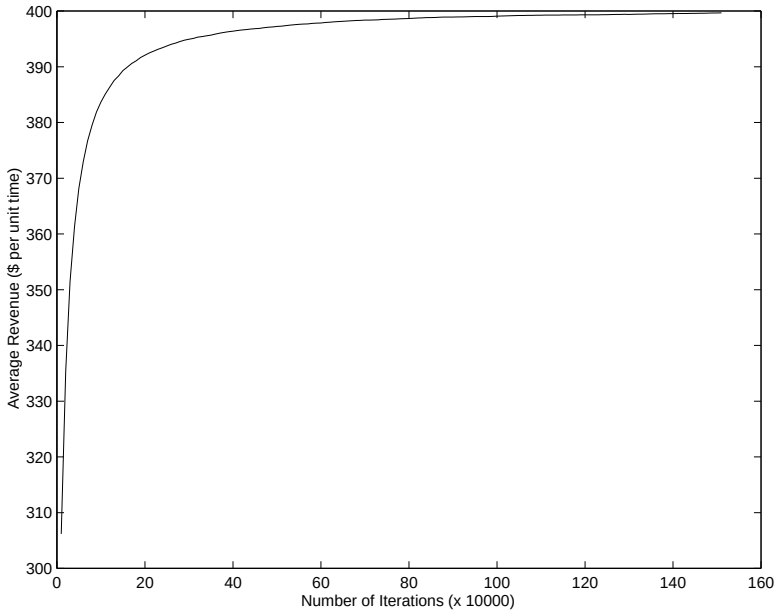

Figure 3: Learning curve for Case 3

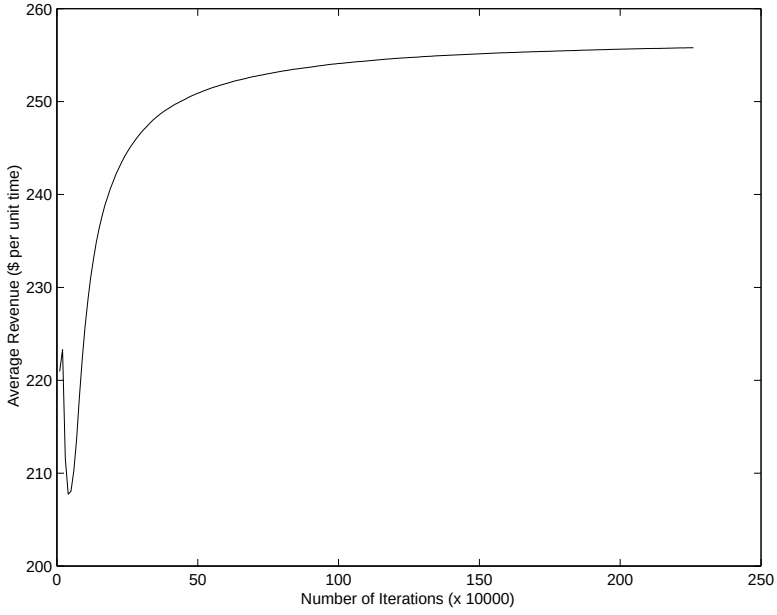

Figure 4: Learning curve for Case 5

6 Results

Shown in Figure 3 and Figure 4 are sample learning curves (using λ-SMART) for Cases 3 and 5 of the sample problems presented in Table 1. The results obtained from λ-SMART and the heuristic for all the sample cases that were run under identical considerations are shown in Table 2. The results were averaged over 35 runs, where each simulation run lasted for 1 million time units (several thousand flights). In Table 2, ρ_{RL} and ρ_{EMSR} denote respectively the revenue generated (in \$) per unit time by using the RL policy and the EMSR heuristic policy (\$/time). The 95% confidence intervals on the revenues are also shown. It may be noted that the 95% confidence intervals for ρ_{RL} and ρ_{EMSR} did not overlap for any of the 20 test problem cases showing that the RL algorithm significantly outperforms the EMSR policy. The countable part of the state-space of the largest problem considered here is of the order of 10^9.

7 Sensitivity Analysis

Every airline generates estimates of parameters related to passenger arrival from past data. Examples of such parameters are the time between arrival, the probability of cancellation, the percentage of high and the medium class passengers etc. It is of great importance for any airline to fully comprehend the effects that possible errors in the estimates of these parameters may have on the booking policies derived using them, and on the resulting yield. One way to study this is to determine the statistical significance of these parameters with respect to the yield. The input parameters (main factors) that are considered to be important in our sensitivity study are as follows.

- The arrival rate of customers to the system (λ_a)

- The probability of cancellation (p_c^i)
- The probability of a customer belonging to the low fare class (p_l)
- The probability of a customer belonging to the middle fare class (p_m)

Each of the 4 factors listed above were considered at 2 levels, and a full factorial (2^4) experiment was conducted with a single replication. It requires a total of 16 data points. The factors and their levels are shown in Table 3. (It may be noted that to consider the probability of cancellation as one factor, we use only those data points where the probability is identical for all classes. Hence the superscript in p_c^i has been dropped.) The ANOVA (Analysis of variance) performed on the 16 pieces of data obtained from the full factorial experiment is shown in Table 4. From the ANOVA, it can be seen that all the 4 factors are significant with respect to the average revenue generated which indicates that great care must be taken while estimating them. The factor interactions at all levels were found to be insignificant and hence the sum of the squares of the interactions were pooled to obtain an error estimate. The plots of the sensitivity of the average revenue generated for a particular flight with respect to the 4 main factors (one at a time) are shown in Figures 5 through 8. Since the factor interactions are negligible, the one factor at-a-time plots are quite useful in depicting the factor sensitivities. The figures 5 through 8 can be explained as follows. In Figure 5, an RL-based policy from Case 4 is considered for which the estimate of λ_a used is 1.5. The dotted line depicts the performance of RL-based policies derived using the true value of λ_a, whereas the solid curve represents the performance of the policy (which is optimal for $\lambda_a = 1.5$) for different values of λ_a. Hence the difference between the curves is an indicator of the loss of performance due to incorrect estimation of λ_a. As is evident form Figure 5, the loss is an increasing function of the error in estimating λ_a. All the other parameter sensitivities (Figures 6, 7 and 8) are also studied using Case 4. Underestimation of p_c, the probability of cancellation results in greater loss than overestimation over the range of values for which it is studied (see Figure 6). For parameter p_l, overestimation seems to

cause a large loss in revenue while underestimation seems not to make any difference (see Figure 7). The sensitivity of parameter p_m is similar to that of λ_a (see Figure 8).

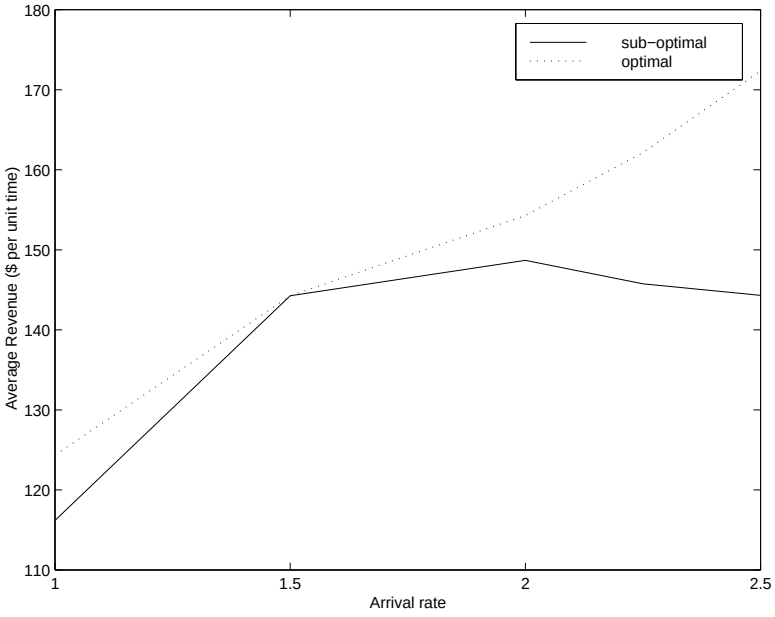

Figure 5: Sensitivity of λ_a

8 Conclusions

This paper considered an important problem faced by the airline industry, namely that of yield management. The problem was formally modeled as an SMDP and subsequently solved using an RL algorithm. Most real life complexities involved in the yield management problem such as overbooking and cancellations were considered in the model thereby making the model a close representative of a real life system. This also made it a complex large-scale stochastic optimization problem whose state-space is of the order of 10^9 (countable part). A contribution of this paper is an extension of an existing algorithm namely SMART to a more fundamental engine of RL, namely that of temporal differences (TD). It was shown that the RL model outperformed a well-known heuristic method that is widely used in the

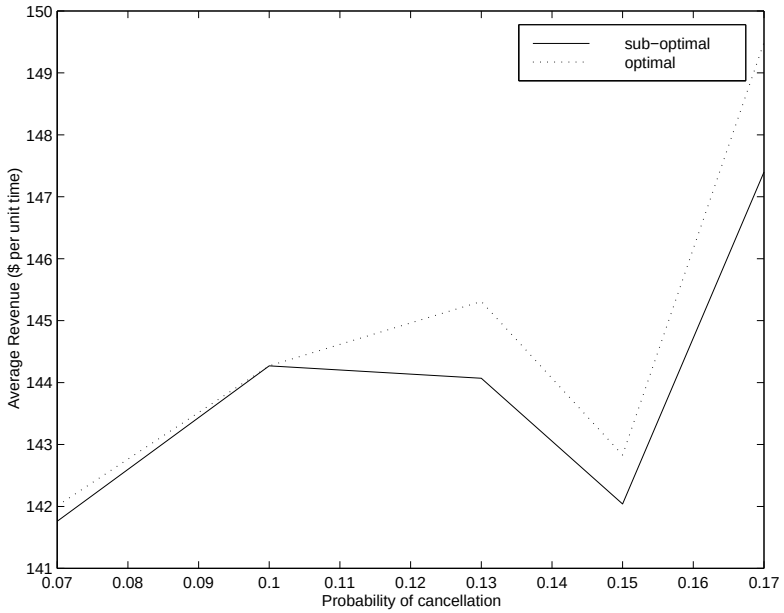

Figure 6: Sensitivity of p_c

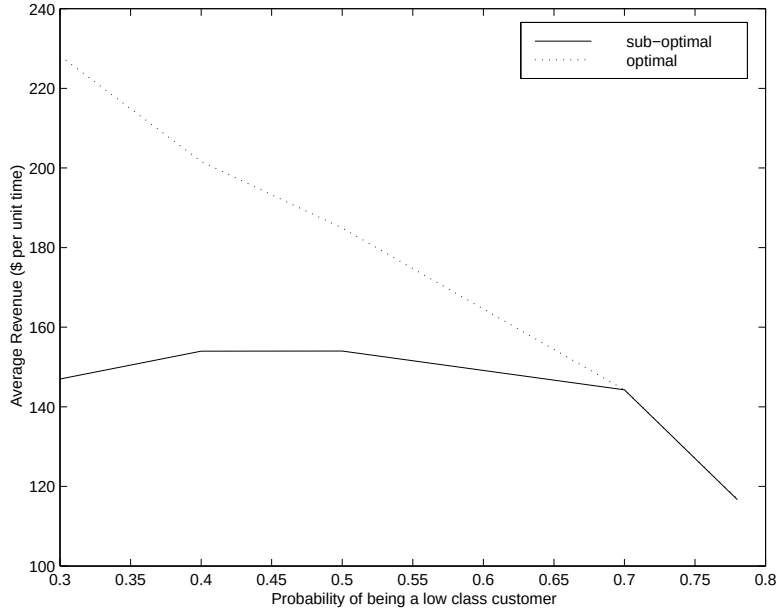

Figure 7: Sensitivity of p_l

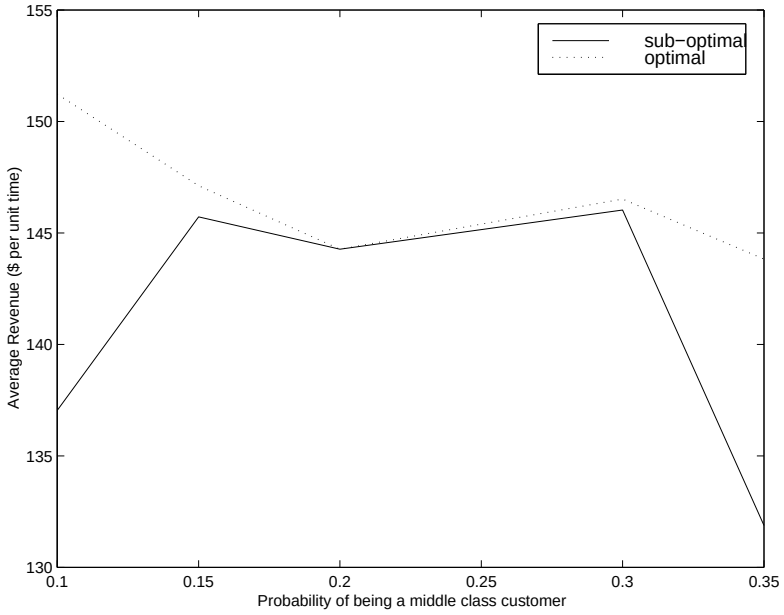

Figure 8: Sensitivity of p_m

industry. It may also be concluded that reinforcement learning and other simulation-based optimization procedures will play a role in providing quality decision support systems for the airline industry in the future - a belief that is echoed in a discussion for suggested directions for furture research in the review paper written by McGill and Van Ryzin [19].

Issues such as block-booking and optimization over multiple legs still remain largely unaddressed in the literature and form exciting topics for further research in this area. An SMDP model hinges on a Markovian assumption. The use of simulation in a path-dependent (non-Markovian) scenario has also attracted research interest (see Higle and Sen [13] and Shapiro [25]). Use of a stochastic programming approach along these lines is also clearly a topic for further research in this area.

References

[1] N. Bandla. Airline yield management using a reinforcement learning approach. Unpublished Master's Thesis, University of South Florida, Tampa, Department of Industrial and Management Systems Engineering, December, 1998.

[2] R. Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc*, 60:503–516, 1954.

[3] P.P. Belobaba. Application of a probabilistic decision model to airline seat inventory control. *Operations Research*, 37:183–197, 1989.

[4] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, 1996.

[5] D.P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts, 1995.

[6] S.L. Brumelle and J.I. McGill. Airline seat allocation with multiple nested fare classes. *Operations Research*, 41:127–137, 1993.

[7] R. E. Chatwin. Multiperiod airline overbooking with a single fare class. *Operations Research*, 46 (6):805–819, 1998.

[8] R.E. Curry. Optimal airline seat allocation with fare classes nested by origins and destinations. *Transportation Science*, 24:193–204, 1990.

[9] C. Darken, J. Chang, and J. Moody. Learning rate schedules for faster stochastic gradient search. In D.A. White and D.A. Sofge, editors, *Neural Networks for Signal Processing 2 - Proceedings of the 1992 IEEE Workshop.* IEEE Press, Piscataway, NJ, 1992.

[10] T. K. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck. Solving semi-markov decision problems using average reward reinforcement learning. *Management Science*, 45(4):560–574, 1999.

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

[12] F. Glover, R. Glover, J. Lorenzo, and C. McMillan. The passenger-mix problem in the scheduled airlines. *Interfaces*, 12, 1982.

[13] J. L. Higle and S. Sen. Stochastic decomposition: an algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16:650–669, 1991.

[14] R. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[15] L. P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning, a survey. *Journal of Artificial Intelligence*, 4, 1996.

[16] K. Littlewood. Forecasting and control of passenger bookings. *In Proceedings 12th AGIFORS Symposium*, pages 95–117, 1972.

[17] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159–196, 1996.

[18] R. Matinez and M. Sanchez. Automatic booking level control. *10th AGIFORS Symposium*, pages 1–20, 1970.

[19] J.I. McGill and Garrett J. Van Ryzin. Revenue management: Research overview and prospects. *Transporation Science*, 33(2):233–256, 1999.

[20] M. L. Puterman. *Markov Decision Processes*. Wiley Interscience, New York, USA, 1994.

[21] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22:400–407, 1951.

[22] L. W. Robinson. Optimal and approximate control policies for airline booking with sequential nonmonotonic fare classes. *Operations Research*, 43:252–263, 1995.

[23] S. M. Ross. *Stochastic Processes*. John Wiley Sons, New York, NY, 1983.

[24] Garrett J. Van Ryzin and J. I. McGill. Revenue mangement without forecasting or optimization: An adaptive algorithm for determining seat protection levels. Working paper at Columbia University, New York.

[25] Alexander Shapiro. Stochastic programming by monte carlo methods. *Preprint, Georgia Institute of Technology*, 2000.

[26] E. Shlifer and Y. Vardi. An airline overbooking policy. *Transportation Science*, 9:101–114, 1975.

[27] B. C. Smith, J. F. Leimkuhler, and R. M. Darrow. Yield management in american airlines. *Interfaces*, 22:8–31, 1992.

[28] J. Subramaniam, S. Stidham Jr, and C.J. Lautenbacher. Airline yield maangement with overbooking, cancellations and no-shows. *Transporatation Science*, 33(2):147–167, 1999.

[29] R. Sutton and A. G. Barto. *Reinforcement Learning*. The MIT Press, Cambridge, Massachusetts, 1998.

[30] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[31] H.R. Thomson. Statistical problems in airline reservation control. *O.R. Quarterly*, 12:167–185, 1961.

[32] D.M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26:305–321, 1978.

[33] R. Wheeler and K. Narenda. Decentralized learning in finite markov chains. *IEEE Transactions on Automatic Control*, 31(6):373–376, 1986.

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

$FS_1 = (100, 175, 250, 300)$,$FS_2 = (199, 275, 350, 400)$, $FS_3 = (350, 450, 550, 600)$,$CD_1 = (0.5, 0.3, 0.2)$ $CD_2 = (0.7, 0.2, 0.1)$,$CD_3 = (0.3, 0.3, 0.4)$					
Case	C	λ_a	CD_i	FS_i	$p_c^i, i = 1, 2, 3$
1	100	1.0	CD_1	FS_1	0.1,0.1,0.1
2	100	1.0	CD_1	FS_2	0.15,0.15,0.15
3	100	1.0	CD_1	FS_3	0.2,0.2,0.2
4	100	1.5	CD_2	FS_1	0.1,0.1,0.1
5	100	1.5	CD_2	FS_2	0.1,0.15,0.15
6	100	1.5	CD_2	FS_3	0.2,0.2,0.2
7	150	2.0	CD_3	FS_1	0.1,0.1,0.1
8	150	2.0	CD_3	FS_2	0.15,0.15,0.15
9	150	2.0	CD_3	FS_3	0.2,0.2,0.2
10	200	2.0	CD_2	FS_1	0.1,0.1,0.1
11	200	2.0	CD_2	FS_2	0.15,0.15,0.15
12	200	2.0	CD_2	FS_3	0.2,0.2,0.2
13	150	2.0	CD_3	FS_2	0.0,0.0,0.0
14	200	2.0	CD_1	FS_1	0.0,0.0,0.0
15	100	2.0	CD_1	FS_1	0.1,0.1,0.1
16	100	2.0	CD_1	FS_2	0.15,0.15,0.15
17	100	2.0	CD_1	FS_3	0.2,0.2,0.2
18	100	1.0	CD_1	FS_2	0.1,0.15,0.2
19	100	2.0	CD_1	FS_3	0.15,0.2,0.25
20	150	2.0	CD_3	FS_1	0.05,0.1,0.15

Table 1: Sample problems

Case	ρ_{RL}	95% C.I.	ρ_{EMSR}	95% C.I.
1	147.73	±0.42	144.42	±1.3
2	243.23	±0.69	238.23	±1.67
3	400.94	±1.08	394.47	±3.78
4	144.27	±0.28	142.13	±0.83
5	257.09	±0.43	251.88	±1.25
6	454.19	±0.57	444.44	±2.48
7	310.68	±0.57	260.74	±1.98
8	476.84	±0.76	455.01	±1.98
9	810.42	±0.9	795.3	±2.79
10	247.83	±0.5	233.64	±1.23
11	443.50	±0.77	424.46	±2.19
12	741.92	±1.46	731.47	±5.16
13	444.70	±0.62	357.88	±1.98
14	293.71	±0.45	289.14	±1.61
15	191.44	±0.35	128.03	±0.76
16	306.66	±0.44	252.10	±1.38
17	525.54	±0.9	470.96	±2.57
18	244.19	±1.93	238.99	±2.44
19	402.84	±2.95	395.78	±3.97
20	301.36	±2.04	293.77	±1.10

Table 2: Results from the Sample Problems

Factor	Level 1	Level 2
λ_a	2.25	2.5
p_c	0.1	0.13
p_l	0.5	0.55
p_m	0.3	0.35

Table 3: Factors and their levels for Sensitivity Analysis

Source of variation	Sums of squares (SS)	degrees of freedom (df)	Variance SS/df	F_0	F_c
λ_a	466.18	1	466.18	106.50	4.84
p_c	260.51	1	260.51	59.51	4.84
p_l	1224.30	1	1224.30	279.71	4.84
p_m	125.66	1	125.66	28.71	4.84
Error	48.15	11	4.37		
Total	2124.75	15			

Table 4: Analysis of variance (ANOVA) Results