Missouri S&T Library and Learning Resources

Scholars' Mine

Masters Theses

Student Theses and Dissertations

Summer 2017

A bounded actor-critic algorithm for reinforcement learning

Ryan Jacob Lawhead

Follow this and additional works at: https://scholarsmine.mst.edu/masters_theses

 Part of the Artificial Intelligence and Robotics Commons, and the Operations Research, Systems Engineering and Industrial Engineering Commons

Department:

Recommended Citation

Lawhead, Ryan Jacob, "A bounded actor-critic algorithm for reinforcement learning" (2017). *Masters Theses*. 7740.
https://scholarsmine.mst.edu/masters_theses/7740

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

A BOUNDED ACTOR-CRITIC ALGORITHM FOR REINFORCEMENT LEARNING

by

RYAN JACOB LAWHEAD

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

2017

Approved by

Dr. Abhijit Gosavi, Co-Advisor
Dr. Susan L. Murray, Co-Advisor
Dr. Zeyi Sun

© 2017
Ryan Jacob Lawhead
All Rights Reserved

ABSTRACT

This thesis presents a new actor-critic algorithm from the domain of reinforcement learning to solve Markov and semi-Markov decision processes (or problems) in the field of airline revenue management (ARM). The ARM problem is one of control optimization in which a decision-maker must accept or reject a customer based on a requested fare. This thesis focuses on the so-called single-leg version of the ARM problem, which can be cast as a semi-Markov decision process (SMDP). Large-scale Markov decision processes (MDPs) and SMDPs suffer from the curses of dimensionality and modeling, making it difficult to create the transition probability matrices (TPMs) necessary to solve them using traditional methods such as dynamic and linear programming. This thesis seeks to employ an actor-critic algorithm to overcome the challenges found in developing TPMs for large-scale real-world problems. Unlike traditional actor-critic algorithms, where the values of the so-called actor can either become very large or very small, the algorithm developed in this thesis has an updating mechanism that keeps the values of the actor's iterates bounded in the limit and significantly smaller in magnitude than previous actor-critic algorithms. This allows the algorithm to explore the state space fully and perform better than its traditional counterpart. Numerical experiments conducted show encouraging results with the new algorithm by delivering optimal results on small case MDPs and SMDPs and consistently outperforming an airline industry heuristic, namely EMSR-b, on large-scale ARM problems.

ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Abhijit Gosavi and Dr. Susan L. Murray, for their guidance and encouragement throughout my graduate program. I am also appreciative of them for instilling in me the qualities required to undertake academic research. I also wish to thank Dr. Zeyi Sun for serving on my committee.

I am also grateful for the support and funding for my research from both the Intelligent Systems Center at Missouri University of Science and Technology and the Department of Engineering Management and Systems Engineering at Missouri University of Science and Technology.

Finally, I would like to thank my parents for the encouragement and support they have offered.

TABLE OF CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGMENTS iv

LIST OF ILLUSTRATIONS vi

LIST OF TABLES vii

SECTION

1. INTRODUCTION 1
2. PROBLEM DESCRIPTION 4
   - 2.1. MARKOV DECISION PROCESSES (MDP) 4
   - 2.2. SMDP 4
   - 2.3. BELLMAN EQUATION 5
   - 2.4. AIRLINE REVENUE MANAGEMENT 6
     - 2.4.1. Overview 6
     - 2.4.2. EMSR-b 9
     - 2.4.3. SMDP in the Context of ARM. 10
3. LITERATURE REVIEW 12
4. SEMI-MARKOV ACTOR-CRITIC ALGORITHMS FOR AVERAGE COST 15
   - 4.1. TRADITIONAL ACTOR-CRITIC 15
   - 4.2. PROPOSED BOUNDED ACTOR-CRITIC 16
   - 4.3. STEPS IN ALGORITHM 17
5. NUMERICAL RESULTS 19
   - 5.1. SMALL MDP CASES 19
   - 5.2. SMALL SMDP CASES 21
   - 5.3. AIRLINE REVENUE MANAGEMENT CASE STUDY 23
     - 5.3.1. Set Up 23
     - 5.3.2. Experimental Results 24
6. CONCLUSION 45

BIBLIOGRAPHY 46

VITA 49

LIST OF ILLUSTRATIONS

Page

Figure 2.1 Hub and Spoke (Origin- Destination) Airline Network Diagram 8
Figure 4.1 Mechanics of actor-critic algorithm 18
Figure 5.1 Values of V(1) in learning phase for MDP Case 1..... 21
Figure 5.2 Values of V(2) in learning phase for MDP Case 1..... 21
Figure 5.3 Values of V(1) in learning phase for SMDP case 1 23
Figure 5.4 Values of V(2) in learning phase for SMDP case 1 23
Figure 5.5 4-fare-class comparison of results for EMSR-b and Actor-Critic..... 27
Figure 5.6 6-fare-class comparison of results from EMSR-b and Actor-Critic..... 28
Figure 5.7 Learning phase graph for Case 1 in 4-fare-class system 38
Figure 5.8 Learning phase graph for Case 2 in 4-fare-class system 38
Figure 5.9 Learning phase graph for Case 3 in 4-fare-class system 38
Figure 5.10 Learning phase graph for Case 4 in 4-fare-class system 39
Figure 5.11 Learning phase graph for Case 5 in 4-fare-class system 39
Figure 5.12 Learning phase graph for Case 6 in 4-fare-class system 39
Figure 5.13 Learning phase graph for Case 7 in 4-fare-class system 40
Figure 5.14 Learning phase graph for Case 8 in 4-fare-class system 40
Figure 5.15 Learning phase graph for Case 9 in 4-fare-class system 40
Figure 5.16 Learning phase graph for Case 10 in 4-fare-class system 41
Figure 5.17 Learning phase graph for Case 1 in 6-fare-class system 41
Figure 5.18 Learning phase graph for Case 2 in 6-fare-class system 41
Figure 5.19 Learning phase graph for Case 3 in 6-fare-class system 42
Figure 5.20 Learning phase graph for Case 4 in 6-fare-class system 42
Figure 5.21 Learning phase graph for Case 5 in 6-fare-class system 42
Figure 5.22 Learning phase graph for Case 6 in 6-fare-class system 43
Figure 5.23 Learning phase graph for Case 7 in 6-fare-class system 43
Figure 5.24 Learning phase graph for Case 8 in 6-fare-class system 43
Figure 5.25 Learning phase graph for Case 9 in 6-fare-class system 44
Figure 5.26 Learning phase graph for Case 10 in 6-fare-class system 44

LIST OF TABLES

Page

Table 5.1 Results for Value Iteration and Gosavi (2014a) actor-critic on MDP cases..... 20
Table 5.2 Results for Value Iteration and proposed actor-critic on MDP cases..... 20
Table 5.3 Data for SMDP Cases..... 22
Table 5.4 Results for policy iteration and actor-critic on SMDPs..... 22
Table 5.5 Fares in dollars for 4-fare-class and 6-fare-class problems..... 25
Table 5.6 Input Parameters for 4-fare-class and 6-fare-class systems..... 25
Table 5.7 Results for the 4-fare-class systems..... 26
Table 5.8 Results for the 6-Fare-class systems..... 27
Table 5.9 Booking limit results for EMSR-b for 4-fare-class system..... 28
Table 5.10 Booking limit results for EMSR-b for 6-fare-class system..... 29
Table 5.11 Action selection for Class 1 in 4-fare-class systems..... 29
Table 5.12 Action selection for Class 2 in 4-fare-class system..... 30
Table 5.13 Action selection for Class 3 in 4-fare-class system..... 30
Table 5.14 Action selection for Class 4 in 4-fare-class system..... 31
Table 5.15 Action selection for Class 1 in 6-fare-class system..... 32
Table 5.16 Action selection for Class 2 in 6-fare-class system..... 33
Table 5.17 Action selection for Class 3 in 6-fare-class system..... 34
Table 5.18 Action selection for Class 4 in 6-fare-class system..... 35
Table 5.19 Action selection for Class 5 in 6-fare-class system..... 36
Table 5.20 Action selection for Class 6 in 6-fare-class system..... 37

1. INTRODUCTION

Markov Decision Problems or Processes (MDPs) are problems of sequential decision making in which Markov chains dictate the system's dynamics and behavior. In every state visited by the system, a decision must be selected from the set of permitted actions in that state. The objective considered in this thesis is to maximize a given cost or reward function over an infinite period of time, or time horizon; an infinite time horizon is chosen when one seeks to observe the system for a long time period and the system settles down into a steady state after a long period of time.

In MDPs, the time of transition from one state to another is the same for every jump and is considered to be one unit of time for every transition. MDPs are a special case of what is called a Semi-MDP (SMDP). A SMDP has transition time explicitly modeled into the objective function as a random variable. SMDPs have applications in numerous fields including queuing control (Sennott, 1999), supply chain management (Buffett and Scott, 2004), and maintenance management (Schouten and Vanneste, 1995).

Classical methods of solving both MDPs and SMDPs are dynamic programming (DP) and linear programming (LP). DP seeks to solve these problems using so-called transition probabilities (TPs). The TP is the probability of transitioning from one state to another under a given action permitted in that state. Because TPs are required in DP, the latter tends to breakdown when the number of state-action pairs exceeds a few thousand. Once a sufficiently large number of state-action pairs are reached, the so-called transition probability matrices (TPMs) become too large or complex to compute, especially on large-scale problems in the real world.

A system containing too many state-action pairs exhibits the *curse of dimensionality*, while the challenge of large complexity is known as the *curse of modeling*. A system with n states and m actions would yield a $n \times n$ TPM for each of the m actions or $n \times n \times m$ elements. As a result, when n and m are large, it is difficult to store and process all the elements of the TPMs, and then the curses of dimensionality and modeling set in. LP requires that the number of constraints equals the number of state-actions pairs causing it to exhibit the curse of dimensionality as well. With the need to model more

complex problems and the significant complexity involved in defining TPMs, Reinforcement Learning (RL) was born.

The MDP was developed by Bellman (1957) who also formulated what is now known as the Bellman optimality equation that serves as the foundation for the more modern Reinforcement Learning (RL) algorithms. RL is a simulation-based technique that seeks to solve MDPs and SMDPs when the TPM becomes too large or too complex to compute. By utilizing discrete-event simulations of the system, RL allows us to bypass the construction of the TPMs and thus avoids the curses of dimensionality and modeling while still producing near optimal solutions.

RL algorithms are typically classified into two major categories: Q-learning algorithms and actor critic (adaptive critic) algorithms. Q-learning algorithms are primarily based on value iteration while actor-critic algorithms are based on policy iteration. This thesis will focus on an algorithm rooted in the latter of the two categories.

In this thesis, a new version of the actor-critic algorithm is presented and will be applied to a revenue management problem from the airline industry. The objective of the actor-critic algorithm in this revenue management problem is to maximize the long-run average reward over a given set of flights. This is achieved through exploration of the appropriate number of seats to allocate to each fare class for any given origin-destination path. The algorithm will be compared to a widely-used industrial heuristic known as Expected Marginal Seat Revenue-b (EMSR-b).

The main contributions of this thesis are threefold. First, it modifies the algorithms from the literature (Kulkarni et al. (2011); Gosavi (2014a)) into one whose actor values not only remain bounded but also tend to have small magnitudes, which is numerically very efficient, as these values are used in the algorithm as powers of an exponential term when selecting actions. Second, the modified algorithm leads to a more thorough exploration of the state-action space, which in turn lowers the probability of sub-optimality in practice. Third, the new algorithm is tested on a large-scale airline revenue management problem with several million states, where it outperforms an industrial-scale heuristic.

The remainder of this thesis is organized as follows: Section 2 provides a background on MDPs and SMDPs, along with the details of airline revenue management and the EMSR-b heuristic. Section 3 reviews the literature on reinforcement learning

techniques as well as on airline revenue management. Section 4 provides an in-depth discussion on actor-critic algorithms for application in solving the SMDP for average reward. Section 5 describes numerical results on small MDP and SMDP cases benchmarked against optimal solutions, an experimental set up, and numerical results for large-scale airline revenue management problems. Section 6 concludes the thesis with closing remarks and a discussion of prospective future work in this field.

2. PROBLEM DESCRIPTION

This section provides a background on Markov and semi-Markov decision processes, as well as on the airline revenue management problem considered in this thesis.

2.1. MARKOV DECISION PROCESSES (MDP)

The MDP consists of five major elements. These elements are:

- A decision maker: Also called the agent or controller, the decision maker selects the actions that can occur in the system.
- Policies: The policy is a n-tuple consisting of the action to be selected in each state of the MDP according to the decision maker.
- Transition probability matrices (TPMs): The TPM is a matrix associated with each action that contains the probability of transitions between states in the system.
- Transition reward matrices (TRMs): The TRM is similar to the TPM but instead of the probability, it contains the immediate reward associated with any given transition under a specific action. A negative reward is equivalent to a cost.
- A performance metric (Objective function): The performance metric is a quantifiable value(s) that is used to measure the performance of the system. Much of the literature on infinite time horizon considers two metrics: the long run average reward and the total discounted reward. Long-run average reward is the expected revenue per unit time calculated over an infinite time period. Total discounted reward is also calculated over an infinite time period but accounts for the time value of money.

2.2. SMDP

In the SMDP, the goal is to determine the best action to execute in each state when the time of transition from one state to another is a random variable, which is also considered in the objective function. In this thesis, the long-run average reward objective function will be used due to the short time periods involved in airline revenue management. First, some notation is required prior to defining long-run average reward.

- S: the finite set of states
- $A(i)$: the finite set of actions permitted in state i
- $\mu(i)$: the action chosen in state i when policy μ is pursued, where $\cup_{i\in S} A(i) = A$
- $r(i,a,j)$: one-step immediate reward of transition from state i to j under action a
- $t(i,a,j)$: time spent in one transition from state i to j under action a
- $p(i,a,j)$: probability associated with the transition from state i to j under action a

It follows that the expected immediate reward earned in state i when action a is chosen is defined as: $\bar{r}(i,a) = \sum_{j=1}^{|S|} p(i,a,j)r(i,a,j)$, and the expected transition time is defined as $\bar{t}(i,a)=\sum_{j=1}^{|S|} p(i,a,j)t(i,a,j)$. Now the long-run average reward can be expressed as follows:

Let

$$R(i) = \lim_{k\to\infty} \frac{\mathrm{E}[\sum_{s=1}^{k} \bar{r}(x_s,\mu(x_s))|x_1=i]}{k} \quad (1)$$

and

$$T(i) = \lim_{k\to\infty} \frac{\mathrm{E}[\sum_{s=1}^{k} \bar{t}(x_s,\mu(x_s))|x_1=i]}{k}. \quad (2)$$

Then the long-run average reward of a policy μ in a SMDP, starting at any state i, is

$$\rho_\mu = \frac{R(i)}{T(i)}. \quad (3)$$

2.3. BELLMAN EQUATION

The objective of the average reward SMDP is to find a policy μ that maximizes the reward ρ_μ. Traditionally, this is done using the Bellman optimality equation for SMDPs. The traditional Bellman equation for SMDPs under average reward is presented below:

$$V(i) = \max_{a\in A(i)} \left[\bar{r}(i,a) - \rho^* \bar{t}(i,a) + \sum_{j=1}^{|S|} p(i,a,j)V(j)\right] \; for\ all\ i \in S. \quad (4)$$

The optimal average reward will be denoted by ρ^* throughout this thesis. Equation 4 above implies that if the solution to vector V and the scalar of ρ^* can be found, then the following policy, *d*, is optimal, where

$$d(i) = \underset{a\in S}{argmax} \left[\bar{r}(i,a) - \rho^* \bar{t}(i,a) + \sum_{j=1}^{|S|} p(i,a,j)V(j)\right] \text{ for all } i \in \text{S}. \quad (5)$$

The RL algorithms studied in this thesis will seek to solve the Bellman optimality equation presented above while bypassing the transition probabilities.

2.4. AIRLINE REVENUE MANAGEMENT

This section provides an overview of the airline revenue management (ARM) problem as well as presents the industry heuristic typically used to solve this problem. It also presents ARM in the context of SMDPs to be used throughout this thesis.

2.4.1. Overview. The airline revenue management problem is a well-studied resource-allocation or inventory-control problem that started gaining significant attention in 1978 with deregulation of the airline industry in the United States. The deregulation gave the airlines flexibility to determine their own schedules, routes and fares, as long as they followed FAA (Federal Aviation Administration) guidelines. More recently with the progress of DP and simulation, "it has become possible to study the problem using near-optimal or optimal techniques" (Kulkarni et al. , 2011).

The airline revenue management problem is essentially an inventory-control problem in which the decision maker must decide whether to accept or reject customers as they arrive, via a website (McGill and van Ryzin, 1999). Durham (1995) estimates that a reservation system may need to handle up to five thousand potential bookings per second. The customer in the main cabin of the economy class (business class or higher classes here are not considered here) is generally offered a set of several different fares for a given origin-destination plan. Internally, for the airline, each fare is associated to a fare class. Different fare classes do not imply that the seats are located in different sections of the plane; all seats are available to all fare classes within the cabin. As more customers arrive, generally, the lower fare classes are gradually closed down by the airline. This is because in general the lower fare classes have the greatest demand and are sold first; however, do note that the higher fare classes may offer advantages, and hence *some* passengers who arrive early in the booking horizon may actually buy higher fares even when lower fares are still available. Customers who choose to pay a higher fare, even when lower ones are available, typically receive better benefits such as a lower cancellation penalty or the ability to board the flight sooner.

Customers arriving earlier in the booking horizon are more likely to get a lower priced ticket. Each airline typically updates its price offerings regularly based on time remaining until departure, preferences of the customer, and many other factors. Prices

have to be adjusted in a suitable manner in order for the continued success of an airline company.

Essentially the problem of setting prices is one of determining the number of seats to be allocated to each fare class to ensure a couple of objectives. The first objective is that all customers do not purchase the cheapest tickets, which would lead to lower profits. The second is to ensure that too many seats are not allocated to higher fare classes, leaving empty seats at the time of departure. Further, airline seats are a perishable commodity meaning that as soon as a flight departs, any empty seats signify a loss in potential revenue; thus, it is necessary to get the arithmetic right in terms of how many seats are sold at each fare, prior to flight departure.

The revenue management problem can be studied in two forms: single-leg and network (Talluri and van Ryzin, 2005). The single-leg version of the problem is the problem being studied in this thesis; it involves a direct flight from one location to another without any layovers. At the single-leg level, each fare is referred to as a class and the problem is one of finding the number of customers allowed to book seats in each fare class. The network version consists of layovers and multiple legs in the flight plan. In the network version, each itinerary-fare combination is referred to as a product and the problem is one of finding the number of customers allowed to book in each given product. Product allocation of one leg in the network problem affecting the allocation and availability of one or more of the other legs leads to increased complexity of analysis in the network problem (Gosavi, 2007).

Although customer-classification factors differ from airline to airline, all airlines use two primary factors: time of the booking request and passenger itinerary. For the following discussion, only 3-fare-classes will be considered, although real-world problems can easily have up to ten fare classes:

- Lowest fare class (LC)—fare class with the lowest ticket price
- Middle fare class (MC) —fare class with ticket prices in between the lowest and highest prices
- Highest fare class (HC)— fare class with the highest ticket price

Time of booking request: Typically, passengers who book earlier in the booking horizon get access to lower fare classes and those who arrive later to higher fare classes. If

classification were carried out on just this factor, then the assumption could be made that the first few customers to book make up the LC while the next set of customers make up the MC and the customers to arrive last would make up the HC, but this is not really what happens in the real world. This is because of the origin-destination issue and also cancellation privileges.

Itinerary: To see how an origin-destination based (itinerary based) classification works Figure 2.1 must be examined. Consider the following itineraries:

- Los Angeles- Las Vegas- Denver- Kansas City
- Seattle- Denver- Kansas City
- Salt Lake City- Denver- Kansas City
- Denver- Kansas City

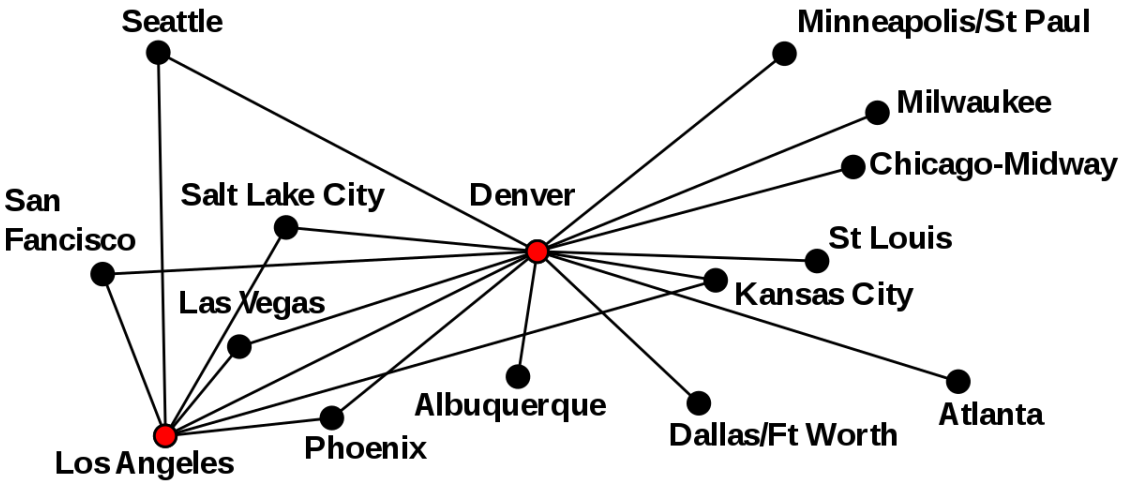

Figure 2.1 Hub and Spoke (Origin- Destination) Airline Network Diagram
Source: https://en.wikipedia.org/wiki/Spoke%E2%80%93hub_distribution_paradigm#/media/file:Airline_hub-1995.svg

Analyzing the fare classes on the single-leg flight from Denver to Kansas City. The passengers originating from Denver and flying to Kansas City would be members of the HC. Passengers from Salt Lake City or Seattle or Las Vegas to Kansas City would form the MC. Passengers from Los Angeles to Kansas City form the LC (Gosavi, 2004).

The pricing and seat allocation problem is "further complicated by the effects of cancellations, no shows, and overbooking" (Gosavi et al, 2007). Each customer that is accepted and purchases a ticket has some probability of canceling their reservation or no

showing. Those passengers in the lower fare class typically have a low probability of cancellation with a high fee for cancellation, while the higher fare classes have a higher probability of cancellation and a low fee. A no-show means that at the time of takeoff the customer had not cancelled but failed to show up to board the aircraft. While the airline keeps all of the ticket fare for the empty seat, they still experience this as a lost opportunity, because they are flying with an empty seat. Hence, airline companies strive to reduce the probability of empty seats; of course, it's a world of cut-throat competition, and every opportunity to make revenues will be seized upon the competitors, making it essential for every airline to ensure that it loses no opportunity to make revenues in a legal manner.

These cancellations and no shows are accounted for by overbooking a flight. This means that the airline company sells more seats than the total number of seats available on the plane. In addition, there are passengers called standby passengers as well. By overbooking, they are attempting to fill any empty seats but they also introduce more risk into the system. If the number of passengers who show up exceed the capacity, then they must pay the passengers who could not get a seat a compensation fee and also find a new flight for that passenger.

2.4.2. EMSR-b. The problem described above can be solved heuristically via the so-called Expected Marginal Seat Revenue (EMSR) rule, which is rooted in Littlewood's equation (Littlewood (1972); Kulkarni et al (2011)). It has two versions: EMSR-a and EMSR-b. The more widely used version in the airline industry is EMSR-b and will therefore be the heuristic used to benchmark the actor-critic algorithm in this thesis. EMSR-b is known to be robust and also capable of producing near-optimal solutions (Belobaba, 1992).

Note that f_i denotes the fare for the ith class and $f_1 < f_2 < f_3 < \cdots < f_n$. Now,

$$\bar{Y}_i = \sum_{j=\mathrm{i}}^{n} Y_j \tag{6}$$

will denote the sum of the demands of all fare classes above and including i, where Y_j denotes the demand in the jth class. The aggregate revenue for the ith class is defined as follows:

$$\bar{f}_i = \frac{\sum_{j=1}^{n} f_j \mathbf{E}[Y_j]}{\sum_{j=1}^{n} \mathbf{E}[Y_j]} \tag{7}$$

Littlewood's equation (Littlewood, 1972) is given as:

$$\bar{f}_i = \bar{f}_{i+1}\Pr(\bar{Y}_{i+1} > P_{i+1}) \quad (8)$$

for $i = 1,2,\ldots,n-1$, where P_i is the so-called protection level for class i. The protection level is the number of seats to be protected for all higher classes from the lower fare classes. For example, P_2 is the number of seats to be protected from class 1 for classes $2, 3, \ldots, n$. There is no protection level for class 1, as it is the lowest fare class from which no protection is needed. The booking limit for the ith class that the airline uses is then defined as:

$$BL_i = \max\{C - P_{i+1}, 0\} \quad (9)$$

for $i = 1,2,\ldots,n-1$. The booking limit for the highest fare class n should clearly be the capacity of the plane if no cancellations occur. However, cancellations can be incorporated into the Littlewood's equation as follows: Replace C in the above by $\frac{C}{1-p}$, where p is the mean cancellation probability over all fare classes (see Kulkani et al. (2011) for additional details). In the above, $(1-p)$ is the so-called correction factor. For solving Equation 9, one must know the distribution of each random variable $\bar{Y}_i$.

2.4.3. SMDP in the Context of ARM. The SMDP for solving the Airline Revenue Management (ARM) problem will be presented in this section. In reality, the ARM problem has a finite horizon, but it can be transformed into one with an infinite time horizon in which the booking horizon is continually reset. Using the infinite time horizon makes it easy to use the RL algorithm discussed in this thesis that is designed for the infinite time horizon. Some notation is required prior to defining the SMDP in context of ARM (which is consistent with that of prior literature, e.g., Kulkarni et al. (2011)):

- s_i: the number of seats sold in fare class i
- n: the number of fare classes
- c: class of the current customer
- t: the time remaining for the departure of the plane
- H: the length of the booking horizon
- Y_i: the demand for the number of customer in class i
- C: the capacity of the plane
- Λ: Poisson rate of arrival of all customer

The objective is to maximize average reward per unit time. The set of actions for this problem contains 2 actions, which are $(Accept, Reject)$, and the state space is as follows:

$$(c, t, s_1, s_2, \dots, s_n, \psi_1, \psi_2, \dots, \psi_n)\ ,$$

where ψ_n is a n-tuple of size s_i that contains the times of arrival (in the booking horizon) of the passengers in the ith fare class.

Due to the size of the state space in this problem, the number of seats sold and arrival times must be encoded using basis functions. The following function from Gosavi (2004) will be used to transform the state space to one with a manageable size for exploration:

$$\varphi = \sum_{i=1}^{n} (s_i \times f_i)/\theta \tag{10}$$

where f_i is the fare rate for the ith class and θ is a user-defined scaling value. The value of θ must be determined through experimentation (trial and error), and its value will be case dependent. This equation actually produces a continuous state space since φ is a continuous variable, but a discrete state space is needed for this algorithm to function. This is handled by rounding the value of φ down to the nearest integer to create a discrete state space. By utilizing this basis function, the state space can now be defined as (c, φ).

3. LITERATURE REVIEW

Traditional methods of solving MDPs and SMDPs use classical DP algorithms such as value iteration (Bellman, 1957) and policy iteration (Howard, 1960). Value iteration is an iterative process that begins with setting the value function for all states to an arbitrary value and then uses the Bellman optimality equation to update and improve the value function. This is repeated until the values calculated on two consecutive iterations fall within a predefined threshold value.

While value iteration explores the state space, policy iteration explores the policy space. Beginning with an arbitrary initial policy, policy iteration requires two stages: the policy evaluation stage and the policy improvement stage. The Bellman equation for a policy is then used to obtain a value function for the current policy. The value function of the current policy is then used to find a better policy (Gosavi, 2014b). These two stages repeat until the value function obtained cannot yield a better policy.

Both of these DP methods depend on the TPs, which in large real-world problems can be difficult to compute. In the absence of the TPs, DP breaks down and, oftentimes, the MDP model is not employed; instead heuristics are used in practice. The root cause for this is that as the problems get more complex, the MDP model is harder to construct for the reasons stated previously, i.e., the curses of dimensionality and modeling. While the benefit of heuristics is that they are simpler to model, most heuristics also provide a lower quality solution than models that utilize MDPs. Therefore, if a problem can be cast as an MDP without the need for TPs, it follows that a lower level of modeling effort would be required while maintaining a high solution quality (Gosavi, 2014b). Reinforcement Learning (RL) seeks to do exactly this.

The benefit of RL methods is seen in that it does not require TPs, implying that it can be used to model problems with much larger state spaces than DP (Bertsekas and Tsitsiklis, 1996). RL deals with four main elements: a policy, a reward function, a value function, and a simulator of the environment. The policy describes the decision maker's behavior in any given state. The reward function expresses the overall objective of the decision maker and guides the decision maker toward an optimal solution. The value function indicates the utility of a state-action pair over the time horizon of the simulation.

The simulation determines the next state given the current state and the selected action (Sutton and Barto, 1999).

RL techniques have been widely applied to problems in supply chains (Pontrandolfo et al. (2002); Chaharsooghi, Heydari, and Zegordi (2008)), manufacturing (Wang and Usher, 2004), and preventative maintenance (Das, et al., 1999). RL typically takes two different paths to solving SMDP and MDP problems through simulation: Q learning (Watkins, 1989), which follows value iteration, and actor-critics (Barto, Sutton, and Anderson (1983); Venayagamoorthy, Harley, and Wunsch (2002)), which follows policy iteration.

This thesis will deal with the actor-critic or adaptive-critic algorithms within the field of RL. Actor-critic algorithms are comprised of two key elements: the *actor* and the *critic*. "The actor is an agent that seeks all potential actions in each state visited," while the "critic is a less reactive agent that updates only when it sees a sensible action from the actor." (Gosavi, 2014a).

Although the convergence of traditional actor-critic algorithms to the optimal solution can be proven mathematically, the actor's values become unbounded in practice. Konda and Borkar (1999) explain "the unboundedness phenomenon by constraining the actor's values in their algorithm" (Gosavi, 2014a); this artificial constraining is achieved via a projection. Their research seeks to overcome this problem of unboundedness and the necessity to artificially constrain the actor's values. Gosavi (2014a) presents a method of reining in the actor value that eliminates the need to artificially constrain the values but only considers small case MDPs. This thesis will apply a new version of the actor-critic algorithm that is different from the one in Konda and Borkar (1999) and the one in Gosavi (2014a) to a problem from the airline industry. Kulkarni et al. (2011) was an extension of the MDP algorithm in Konda and Borkar (1999) to SMDPs, but still required the artificial projection of Konda and Borkar (1999).

With the deregulation of the airline industry in 1978, revenue management (RM) and optimization techniques began gaining popularity. By using RM techniques, American Airlines estimated that it generated $1.4 billion in additional incremental revenue over a three-year period around 1988, and many other airlines reported similar revenue growth due to RM practices (Talluri and van Ryzin, 2005).

As discussed in Section 2.3, the airline industry widely uses the EMSR heuristic that has two version EMSR-a and EMSR-b. Both versions are rooted in Littlewood's equation (Littlewood, 1972) and use approximation to condense the problem at each stage to two classes: the current class and all classes above that class (Talluri and van Ryzin, 2005). EMSR-a (Belobaba, 1989) is based on the aggregation of protection levels, while EMSR-b aggregates the demand (Belobaba (1992); Talluri and van Ryzin (2004)). Both of these heuristics are known to produce near-optimal results in perfect conditions, but have been found to lose reliability once cancellations, no shows, and overbookings are considered. To overcome this limitation, RL algorithms can be applied to solve the ARM problem.

More recently with the advancements in simulation, it has become possible to model this problem using RL. While the λ-SMART algorithm (Gosavi, Bandla, and Das, 2002) and the actor-critic algorithm in Kulkarni et al. (2011) are RL algorithms that have been applied to the airline revenue management problem in the past, the actor-critic algorithms of the past were unstable and became unbounded in certain situations; further the λ-SMART algorithm is based on a finite trajectory, which may not be applicable to all RL settings. This thesis considers a new algorithm that provides a more stable and robust solution while still outperforming the industry standard, namely the EMSR-b heuristic.

4. SEMI-MARKOV ACTOR-CRITIC ALGORITHMS FOR AVERAGE COST

Actor-critics or adaptive critics are well-studied algorithms within the RL family (Werbos (1987); Venayagamoorthy, Harley, and Wunsch (2002)). "The *actor* is an agent that seeks all possible actions in each state visited, while the *critic* is a less reactive agent that updates only when it sees a sensible action from the actor" Gosavi (2014a). Actor-critics are advantageous in that they can solve the SMDP via simuation without computing the transition probabilities (Kulkarni et al, 2011). Actor-critics are based on the policy iteration algorithm rather the value iteration algorithm, which has led to Q-Learning (Watkins, 1989).

The remainder of this section discusses the traditional actor-critic algorithm as well as our proposed new algorithm. Subsection 4.1 present the traditional actor-critic algorithm, and in Subsection 4.2, the new algorithm is discussed. Subsection 4.3 presents the step-by-step details of the new algorithm.

4.1. TRADITIONAL ACTOR-CRITIC

As discussed above, the actor-critic algorithm for the MDP has two main elements: an *actor* that selects a policy for each state and a *critic* that computes the value function for each policy. For the actor-critic to be applied to an average reward SMDP, a third step must be added to the algorithm in which the critic also evaluates the average reward of the policy (Kulkarni et al, 2011). This step is necessary due to the added elements of average reward and stochastic transition times. Some additional notation that is needed at this point is defined next:

- $P(i,a)$: Actor value for action a in state i
- $V(i)$: Critic value for state i
- $q(i,a)$: Probability of selecting action a in state i
- η: A tunable contraction factor which should be in the interval (0,1)
- α: A step size for the actor update
- β: A step size for the critic update
- γ: A step size for the average reward update
- ρ: Average reward

The key updates for the SMDP in the traditional version of the algorithm from Kulkarni et al (2011) are as follows:

Traditional Actor Update:

- $P(i,a) \leftarrow P(i,a) + \alpha[r(i,a,j) - \rho t(i,a,j) + \eta V(j) - V(i)]$ (11)
- If $P(i,a) > \bar{P}$, set $P(i,a) \leftarrow \bar{P}$. If $P(i,a) < -\bar{P}$, set $P(i,a) \leftarrow -\bar{P}$. The scalar $\bar{P}$ is a positive number fixed at the start of the algorithm. This update is essentially the projection that was discussed above in order to keep the values of the actor bounded.

Critic Update:

$$V(i) \leftarrow (1-\beta)V(i) + \beta[r(i,a,j) - \rho t(i,a,j) + \eta V(j)] \quad (12)$$

Average Reward Update:

$$R \leftarrow R + r(i,a,j) \quad (13)$$

$$T \leftarrow T + t(i,a,j) \quad (14)$$

$$\rho \leftarrow (1-\gamma)\rho + \gamma R/T \quad (15)$$

Note that the algorithm presented above does contain a tunable contraction factor, η, which is typically set close to 1 but strictly less than 1 (Kulkarni et al, 2011).

4.2. PROPOSED BOUNDED ACTOR-CRITIC

Technically, the problem with the traditional actor-critic (Barto, Sutton, and Anderson (1983); Konda and Borkar (1999); Kulkarni et al. (2011); Lawhead, Gosavi, and Murray (2017)) is that the actor's update is *not* the convex combination that is typically seen in RL or neural network updating and as such one cannot expect boundedness from it:

$$New\_value \leftarrow (1-\alpha)Old\_value + \alpha(feedback). \quad (16)$$

Rather, it is of the following form:

$$New\_value \leftarrow Old\_value + \alpha(feedback). \quad (17)$$

The form seen in RL or neural networks uses a step size, α, to average the old value with current feedback value. As α is decayed with each iteration, the feedback will contribute less to the averaging of the new value. Because the traditional actor update does not multiply the old value by $(1-\alpha)$, the new value will always be larger than the old value and theoretically never become bounded.

Due to this problem, the following actor update is proposed in Gosavi (2014a):

$$P(i,a) \leftarrow (1-\alpha)P(i,a) + \alpha[r(i,a,j) - \rho t(i,a,j) + \eta V(j)]. \quad (18)$$

However, the above also encounters a problem in practice that the actor values, i.e., P(.,.), still become quite large, although they are bounded. This can cause problems in the Boltzmann action selection scheme that will be used within the algorithm. In this thesis, a further refinement of the above is proposed, which is as follows:

$$P(i,a) \leftarrow (1-\alpha)P(i,a) + \alpha[r(i,a,j) - \rho t(i,a,j) + \eta V(j) - V(i)]. \quad (19)$$

Note the vital difference between the proposed algorithm and the algorithm in Gosavi (2014a) is the subtraction of the critic value, $V(i)$. By introducing this value into the actor update, it was discovered that the actor values still remained bounded, but were of significantly smaller absolute values than in bounded actor critic of Gosavi (2014a); this will be shown experimentally in Section 5.1.

In what follows, the details of the new algorithm are presented in a step-by-step format.

4.3. STEPS IN ALGORITHM

Step 1. For all l, where $l \in S$, and $u \in A(l)$, set $V(l) \leftarrow 0$ and $P(l,u) \leftarrow 0$. Set k, the number of state changes or iterations, to 0. Set R, T, and ρ to 0. The algorithm is run for k_{max} iterations, where k_{max} is chosen to be a sufficiently large number.

Step 2. Let the current state be i. Select action a with a probability of

$$q(i,a) = \frac{e^{P(i,a)}}{\sum_{b \in A(i)} e^{P(i,b)}} . \quad (20)$$

Step 3. Simulate action a. Let the next state be j. Let $r(i,a,j)$ be the immediate reward earned in going to j from i under a and $t(i,a,j)$ be the time in the same transition. Set $k \leftarrow k+1$ and update $P(i,a)$ using a step size, α:

$$P(i,a) \leftarrow (1-\alpha)P(i,a) + \alpha[r(i,a,j) - \rho t(i,a,j) + \eta V(j) - V(i)]. \quad (21)$$

Step 4. Update V as follows using a step size, β:

$$V(i) \leftarrow (1-\beta)V(i) + \beta[r(i,a,j) - \rho t(i,a,j) + \eta V(j)]. \quad (22)$$

Step 5. Update R, T, and ρ as follows:

$$R \leftarrow R + r(i,a,j) \quad (23)$$

$$T \leftarrow T + t(i,a,j) \quad (24)$$

$$\rho \leftarrow (1-\gamma)\rho + \gamma R/T \tag{25}$$

Step 6. If $k < k_{max}$, set $i \leftarrow j$ and then go to Step 2. Otherwise, go to Step 7.

Step 7. For each $l \in S$, select $d(l) \in \arg max_{b \in A(l)} P(l,b)$. The policy generated by the algorithm is $\hat{d}$. Stop.

In the above algorithm description, to increase clarity, the subscript, k, has been suppressed in V, ρ, P, and also in the step sizes α, β, and γ. This iterative process can be seen pictorially in Figure 4.1.

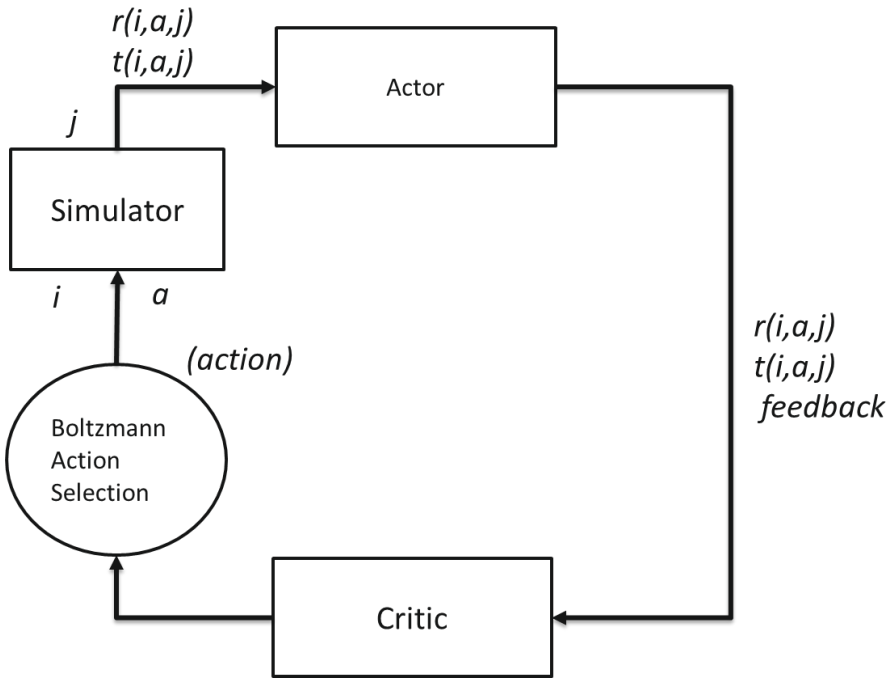

Figure 4.1 Mechanics of actor-critic algorithm

5. NUMERICAL RESULTS

In this section, the experimental results are presented for the proposed actor-critic algorithm. Subsections 5.1 and 5.2 present numerical results with the actor-critic algorithm on small case MDPs and SMDPs, respectively. Section 5.3 presents the airline revenue management case study and results obtained through benchmarking the actor-critic algorithm against an industry-standard heuristic.

5.1. SMALL MDP CASES

The algorithm was run for 4 different discounted reward MDPs consisting of two states each and two actions allowed in each state. Cases have been taken from Gosavi(2014a). The data for each case is as follows, where P_a denotes the TPM for the action a and R_a denotes the TRM for action a. Note that the element in the *i*th column and *j*th row of P_a equals $p(i,a,j)$. Similarly, the element in the *i*th column and *j*th row of R_a equals $r(i,a,j)$.

Case 1:

$$P_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}, \quad R_1 = \begin{bmatrix} 6 & -5 \\ 7 & 12 \end{bmatrix}, \quad R_2 = \begin{bmatrix} 10 & 17 \\ -14 & 13 \end{bmatrix}$$

For the remaining cases, only the values that differ from Case 1 are listed. Also, a discounting factor, $\lambda = 0.8$, was used for all cases.

- Case 2: $r(1,1,2) = 5, r(2,2,1) = 14$;
- Case 3: $r(1,2,1) = 12$;
- Case 4: $r(1,1,1) = 16, r(1,2,1) = 0$.

The algorithm was run for a maximum of 10000 iterations with the following learning rates: $\alpha = \frac{\log(k+1)}{k+1}$, $\beta = \frac{150}{300+k}$. The optimal policy for each case was obtained using Q-value iteration and is denoted as (a_1, a_2), where a_1 denotes the optimal action in state 1 and a_2 denotes the optimal action in state 2.

Table 5.2 shows the optimal policy, μ^*, and the optimal value function, $V^*()$, both obtained from value iteration. Table 5.2 also shows the value function, $V()$, and actor values, $P(i,a)$, obtained from the actor-critic algorithm in Gosavi (2014a). Table 5.2 also shows the value function, $V()$, and actor values, $P(i,a)$, obtained from the proposed actor-

critic algorithm. It can easily be seen that the value functions produced by the actor-critic are essential equal to the optimal values produced from value iteration. *It needs to be pointed out that the values for the actor in Table 5.1 which are based on the algorithm in Gosavi (2014a) have larger absolute values than the ones in Table 5.2, which are from the proposed algorithm.*

The policy produced by the actor-critic can be derived by examining the actor values and finding the action that produces the largest values for each state. For example, in Case 1, $P(1,1) = -8.389$ and $P(1,2) = -0.004$ meaning that action 2 is better in state 1 because $P(1,2) > P(1,1)$. In state 2, $P(2,1) = -0.020$ and $P(2,2) = -3.497$ meaning that action 1 is better in state 2 because $P(2,1) > P(2,2)$. This produces a policy of (2,1), which matches the optimal policy produced from value iteration. Using this methodology, it can be seen that both actor-critic algorithm produces the optimal policy in all 4 cases. By examining the actor values, $P(i,a)$, it can be seen that the proposed algorithm produces values that are significantly small in all cases.

Table 5.1 Results for Value Iteration and Gosavi (2014a) actor-critic on MDP cases

Case	μ*	V(1)	V(2)	V*(1)	V*(2)	P(1,1)	P(1,2)	P(2,1)	P(2,2)
1	(2,1)	52.93	51.68	53.03	51.86	43.95	53.00	52.07	39.48
2	(2,2)	55.38	61.16	55.77	61.45	50.11	55.81	48.48	61.71
3	(2,1)	60.80	56.59	60.83	56.66	49.32	60.81	56.70	43.08
4	(1,1)	49.90	49.35	48.97	49.36	48.87	38.17	49.03	38.75

Table 5.2 Results for Value Iteration and proposed actor-critic on MDP cases

Case	μ*	V(1)	V(2)	V*(1)	V*(2)	P(1,1)	P(1,2)	P(2,1)	P(2,2)
1	(2,1)	52.93	51.68	53.03	51.86	-8.389	-0.004	-0.020	-3.497
2	(2,2)	55.38	61.16	55.77	61.45	-3.887	-0.029	-2.045	-0.036
3	(2,1)	60.80	56.59	60.83	56.66	-10.730	-0.001	-0.001	-3.552
4	(1,1)	49.90	49.35	48.97	49.36	0.070	-7.744	0.022	-1.605

Figure 5.1 and Figure 5.2 show the progression of the value function throughout the first 10000 iterations of the learning phase for V(1) and V(2), respectively for the proposed actor-critic algorithm. It should be noted the value function is plotted every 100

iterations. The red line on each of the graphs indicates the optimal value obtained from value iteration, and it can be seen that the actor critic algorithm is converging to this line.

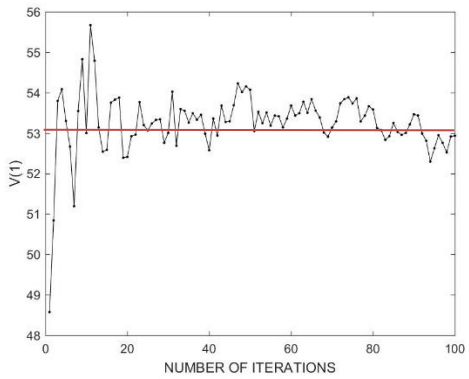

Figure 5.1 Values of V(1) in learning phase for MDP Case 1

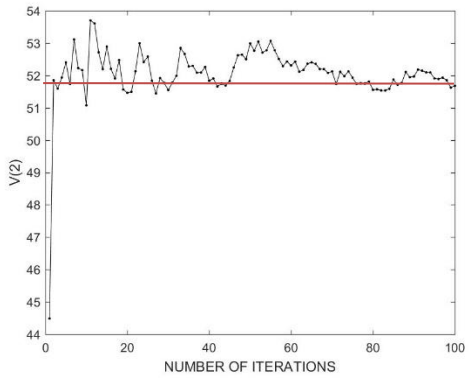

Figure 5.2 Values of V(2) in learning phase for MDP Case

5.2. SMALL SMDP CASES

The actor-critic algorithm was run for 4 different average reward SMDP cases consisting of two states each and two actions allowed in each state. The data for each case is shown in Table 5.3. A value of $\eta = 0.99$ was used for all cases.

The algorithm was run for a maximum of 10000 iterations with the following learning rates: $\alpha = \frac{\log(k+1)}{k+1}$, $\beta = \frac{5}{10+k}$. The optimal policy for each case was obtained using policy iteration and is denoted as (a_1, a_2), where a_1 denotes the optimal action in state 1 and a_2 denotes the optimal action in state 2. Table 5.4 shows the optimal policy, μ^*,

the optimal value function, $V^*(i)$, both obtained from policy iteration. Table 5.4 also shows the value function, $V(i)$, and actor values, $P(i,a)$ obtained from the actor-critic algorithm. Note that unlike in the MDP study, the value functions do not match. This is due to the necessity of setting an arbitrary value function to 0 in policy iteration. Even though the value functions do not match, it can be seen that the policy found by the actor-critic in each case matches the optimal policy by using the same method presented in Section 5.1. Figure 5.3 and Figure 5.4 show the plots of the critic values for the two states, respectively, in the simulation.

Table 5.3 Data for SMDP Cases

Case		$p(i,a,j)$				$t(i,a,j)$				$r(i,a,j)$			
	action a	1		2		1		2		1		2	
	State j →	1	2	1	2	1	2	1	2	1	2	1	2
	State I ↓												
1	1	0.7	0.3	0.9	0.1	1	5	50	75	6	-5	10	17
	2	0.4	0.6	0.2	0.8	120	60	7	2	7	12	-14	13
2	1	0.7	0.3	0.9	0.1	10	5	5	75	6	5	10	17
	2	0.4	0.6	0.2	0.8	120	60	7	20	7	12	14	13
3	1	0.7	0.3	0.9	0.1	10	5	50	75	6	-5	12	17
	2	0.4	0.6	0.2	0.8	12	60	7	20	70	12	6	13
4	1	0.7	0.3	0.9	0.1	10	5	50	75	16	5	80	10
	2	0.4	0.6	0.2	0.8	120	60	7	20	75	120	6	1

Table 5.4 Results for policy iteration and actor-critic on SMDPs

Case	μ*	V(1)	V(2)	V*(1)	V*(2)	P(1,1)	P(1,2)	P(2,1)	P(2,2)
1	(1,2)	33.13	44.06	0.00	6.43	3.832	-3.855	-16.822	2.324
2	(2,2)	51.92	45.73	0.00	-6.71	-1.608	1.006	-28.859	1.786
3	(1,1)	39.85	54.34	0.00	12.09	0.674	-13.192	1.262	-1.979
4	(2,1)	88.92	66.44	0.00	-26.43	-10.115	0.825	-0.907	-6.672

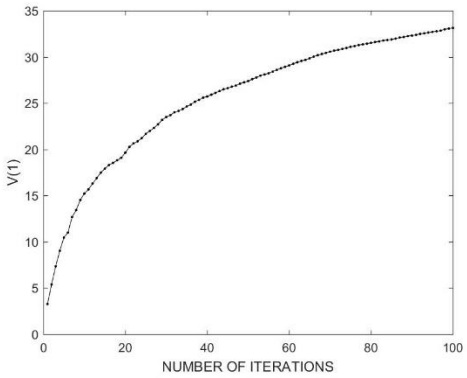

Figure 5.3 Values of V(1) in learning phase for SMDP case 1

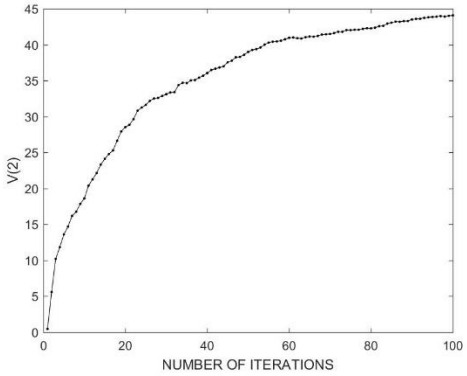

Figure 5.4 Values of V(2) in learning phase for SMDP case 1

5.3. AIRLINE REVENUE MANAGEMENT CASE STUDY

In this section, the numerical results produced by the bounded actor-critic algorithm are presented for the single-leg Airline Revenue Management (ARM) problem, which was discussed in Section 4. Twenty different cases were studied in which a four-fare-class system was used for the first ten cases and a six-fare-class system was used for the following ten cases.

5.3.1. Set Up. The fare structure for each case is given by $FS = (f_1, f_2, f_3, \dots, f_i, b)$, where f_i is the fare of the ith fare class and b is the bumping cost. A lower value of $\boldsymbol{i}$ stands for a lower revenue fare class.

The booking horizon is assumed to be 100 days long, and for the arrivals, a homogeneous Poisson process with a rate $\Lambda = 1.4$ passengers per day is used; the plane is

assumed to have a total capacity of 100 seats. The Poisson process for each fare class is an independent process whose rate equals rate $\Lambda \Pr(i)$, where $\Pr(i)$ denotes the probability that the arrival belongs to class i. The so-called cancellation probability for each fare class is essentially the probability with which a traveler in that given fare class cancels the ticket. When a cancelation occurs, it is scheduled using a uniform distribution between the time of arrival and the time of flight departure. The value of θ is determined separately for each individual case based on careful experimentation to produce the best possible policy. A value of 0.999999 was used for η in the actor-critic algorithm

After significant experimentation, the following step-sizes were used for the three updates in the algorithm:

$$\alpha = \frac{15000}{300000+k}, \qquad \beta = \frac{10000}{300000+3k}, \qquad \gamma = \frac{10000}{300000+10k}.$$

5.3.2. Experimental Results. This section presents the results of the bounded actor-critic algorithm on the airline revenue management problem and compares the result to that from using the well-known industry heuristic, EMRS-b. The performance metric used for comparison of results between the actor-critic algorithm and EMSR-b is "average revenue per day."

The algorithm was tested on 10 cases for each of the four-fare-class systems and the six-fare-class systems. Fares for each case in both systems are shown in Table 5.5, and input parameters are shown in Table 5.6.

The learning phase was run for 1000 flights and took at most 130 seconds on a 64-bit, 2.5 GHz windows operating system in MATLAB. First, the optimal policy is determined; then the simulator is re-run with the fixed policy (also called frozen policy) for 8 replications with 200 flights per replication. Finally, the average rewards per day (ρ) computed from each of the replications were averaged to compute the actual average reward generated by the policy returned the algorithm. This result is denoted as $\rho_{\text{Actor-critic}}$ and the solution for EMSR-b is donated by $\rho_{\text{EMSR-b}}$. A *t*-test using the 8 different replications was performed to determine if the results are delivered from the actor critic differ from those of EMSR-b with 95% confidence in a statistical sense. The algorithms improvement over EMSR-b is defined as:

$$IMP = \frac{\rho_{Actor-critic} - \rho_{EMSR-b}}{\rho_{EMSR-b}} \times 100\ \% \qquad (26)$$

Table 5.5 Fares in dollars for 4-fare-class and 6-fare-class problems

System	Fares (4-fare system)	Fares (6-fare system)
1	75, 200, 400, 550	101, 127, 153, 179, 293, 419
2	80, 200, 400, 500	94, 112, 142, 160, 271, 395
3	75, 150, 300, 550	111, 131, 153, 185, 293, 426
4	80, 150, 400, 550	127, 143, 167, 199, 320, 462
5	70, 150, 350, 550	105, 135, 143, 179, 284, 411
6	125, 180, 225, 400	90, 105, 139, 156, 261, 388
7	100, 175, 250, 400	108, 127, 155, 191, 295, 431
8	100, 150, 200, 450	76, 98, 123, 162, 247, 400
9	119, 139, 239, 430	87, 115, 162, 185, 278, 410
10	145, 209, 280, 350	115, 134, 165, 184, 302, 430

Table 5.6 Input Parameters for 4-fare-class and 6-fare-class systems

Parameter	4-fare-class system	6-fare-class system
Arrival Probabilities	0.6, 0.25, 0.09, 0.06	0.3, 0.3, 0.13, 0.13, 0.09, 0.06
Cancellation Probabilities	0.1, 0.2, 0.2, 0.4	0.1, 0.1, 0.1, 0.2, 0.2, 0.4
Cancellation Penalties Cases 1:5	70, 50, 30, 10	70, 50, 50, 30, 10, 0
Cancellation Penalties Cases 6:10	100, 90, 60, 40	70, 50, 50, 30, 10, 0
Bumping Penalty	200	250

The results for the 4-fare-class system and 6-fare-class system are shown in Table 5.7 and Table 5.8, respectively. Figure 5.5 and Figure 5.6 show the average reward per day comparison between the Actor-Critic and EMSR-b for the 4-fare-class and 6-fare-class systems, respectively. Table 5.9 and Table 5.10 provide the booking limits returned using EMSR-b for the 4-fare-class and 6-fare-class systems, respectively, where $BL(i)$ represent the booking limit of the i^{th} fare class. The optimal policy returned by the actor-critic

algorithm for each case in the 4-fare-class systems is shown Table 5.11– Table 5.14 for fare class 1, 2, 3, and 4, respectively. The optimal policy returned by the actor-critic algorithm for each case in the 6-fare-class systems is shown Table 5.15−Table 5.20 for fare class 1 through 6, respectively. As can be seen from the tables, the Actor-critic algorithm outperforms EMSR-b in a statistically significant manner in each case tested. Figure 5.7−Figure 5.16 show the plots of the average reward for each case in the 4-fare-class systems in the simulation while Figure 5.17−Figure 5.26 show the same plots for the 6-fare-class systems.

Table 5.7 Results for the 4-fare-class systems

Case	$\rho_{EMSR\text{-}b}$	$\rho_{Actor\text{-}critic}$	θ	IMP (%)
1	163.79± 0.515	168.01± 1.454	1400	2.58
2	163.53± 0.365	167.52± 0.895	1200	2.44
3	138.56± 0.241	141.81± 1.195	1500	2.35
4	152.06± 0.417	157.83± 0.958	1900	3.80
5	140.24± 0.350	146.33± 0.438	1500	4.34
6	170.18± 0.403	173.81± 0.530	1800	2.13
7	154.68± 0.427	161.42± 0.489	1500	4.36
8	144.55± 0.651	149.20± 0.447	1000	3.22
9	157.01± 0.446	162.24± 0.253	1100	3.34
10	195.25± 0.421	199.16± 0.365	1700	2.00

Table 5.8 Results for the 6-Fare-class systems

Case	$\rho_{EMSR\text{-}b}$	$\rho_{Actor\text{-}Critic}$	θ	IMP (%)
1	156.86± 0.260	160.28± 0.716	1200	2.18
2	141.16± 0.362	144.03± 0.692	1800	2.03
3	161.08± 0.360	164.68± 0.751	1600	2.23
4	177.89± 0.324	181.01± 0.665	1200	1.75
5	157.09± 0.231	161.06± 0.335	1000	2.53
6	135.55± 0.350	140.50± 0.514	1400	3.65
7	160.84± 0.295	163.01± 0.412	1600	1.35
8	128.02± 0.438	130.63± 0.395	1300	2.04
9	150.41± 0.426	152.84± 0.832	1800	1.62
10	166.01± 0.394	170.11± 0.585	1200	2.47

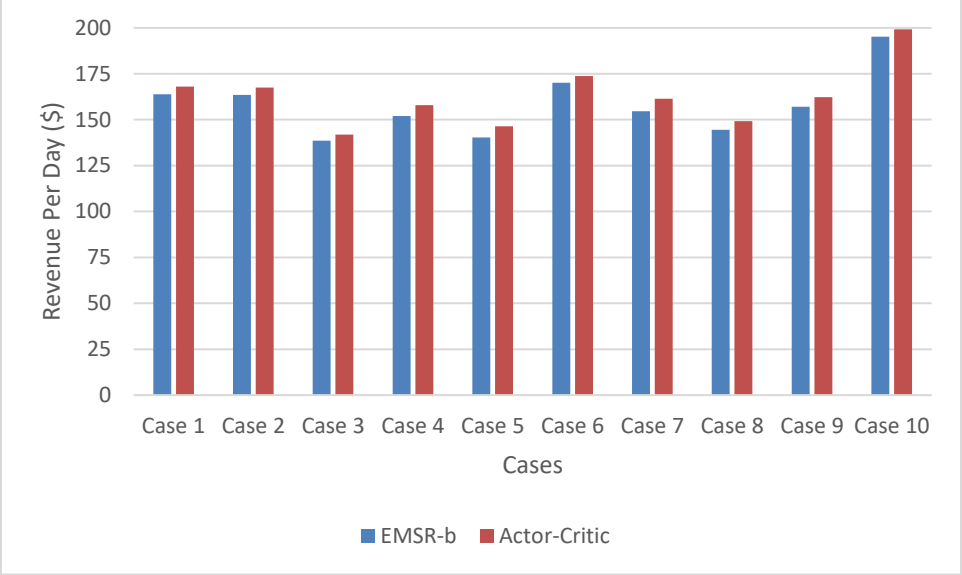

Figure 5.5 4-fare-class comparison of results for EMSR-b and Actor-Critic

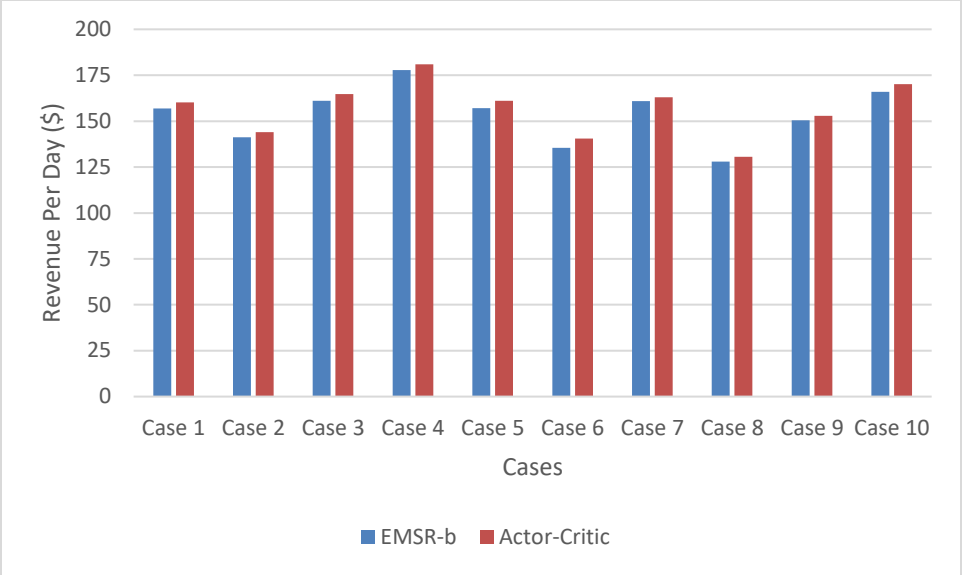

Figure 5.6 6-fare-class comparison of results from EMSR-b and Actor-Critic

Table 5.9 Booking limit results for EMSR-b for 4-fare-class system

Case	BL(1)	BL(2)	BL(3)	BL(4)
1	68	107	122	129
2	69	108	123	129
3	68	107	122	129
4	69	106	122	129
5	69	106	122	129
6	74	109	121	129
7	72	109	122	129
8	73	108	120	129
9	75	107	121	129
10	75	110	123	129

Table 5.10 Booking limit results for EMSR-b for 6-fare-class system

Case	BL(1)	BL(2)	BL(3)	BL(4)	BL(5)	BL(6)
1	25	67	86	103	116	122
2	26	67	86	103	116	122
3	26	68	86	103	116	122
4	27	68	86	103	116	122
5	26	68	85	103	116	122
6	26	66	86	103	116	122
7	26	67	86	103	116	122
8	24	66	85	103	116	122
9	24	66	86	103	116	122
10	27	67	86	103	117	122

Table 5.11 Action selection for Class 1 in 4-fare-class systems

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	r	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	a	a
4	a	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	r	r	r	a	a	a	a	a	a
9	a	r	r	r	r	a	a	a	a	a
10	a	a	a	a	r	r	r	a	a	a
11	a	a	a	r	r	a	r	a	a	a
12	r	a	r	a	a	a	r	a	a	r
13	r	r	a		a	r	r	r	a	r
14	r	a	r					a	r	r
15	r	a						a	a	a
16	r	r						r	r	r
17	r	r						a	a	
18		r						r	a	
19									a	

Table 5.12 Action selection for Class 2 in 4-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	a	a
4	a	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	a	a	a	a	a	a	a	a	a
9	a	a	a	a	a	a	a	a	a	a
10	a	a	a	r	a	a	a	a	a	a
11	a	a	a	a	r	a	a	a	a	a
12	a	a	a	r	a	a	a	a	a	a
13	a	a	a		a	a	r	a	a	a
14	r	a	r					a	a	r
15	a	a						a	r	a
16	a	r						r	r	r
17	r	a						a	r	
18		r						r	r	
19										

Table 5.13 Action selection for Class 3 in 4-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	r	a	a	a	a
4	a	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	a	a	a	a	a	a	a	a	a
9	a	a	a	a	a	a	a	a	a	a
10	a	a	a	a	a	a	a	a	a	a
11	a	a	a	r	a	a	a	a	a	a
12	a	a	a	r	a	a	a	a	a	a
13	a	a	r		a	r	a	a	a	r
14	a	a						a	a	a
15	a	a						a	a	a
16	a	a						a	a	a
17	a	a						a	a	
18		a						a	a	
19										

Table 5.14 Action selection for Class 4 in 4-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	a	a
4	r	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	a	a	a	a	a	a	a	a	a
9	a	a	a	a	a	a	a	a	a	a
10	a	a	a	a	a	a	a	a	a	a
11	a	a	a	a	a	a	a	a	a	a
12	a	a	a	r	a	a	a	a	a	a
13	a	a	a		r	a	a	a	a	a
14	a	a						a	a	a
15	a	a						a	a	a
16	a	a						a	a	a
17	a	a						a	a	
18		r						r	a	
19										

Table 5.15 Action selection for Class 1 in 6-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	r	a
4	a	a	a	a	a	a	a	a	r	a
5	a	a	a	a	a	a	a	a	a	a
6	a	r	a	a	a	a	a	a	r	a
7	a	a	a	a	a	a	a	a	a	a
8	a	r	r	a	r	a	r	a	a	a
9	a	a	a	a	r	a	a	a	r	a
10	a	r	r	r	a	r	a	r	r	a
11	a	r	a	a	r	r	r	r	r	a
12	a		r	a	a	a	r	r		a
13	r		a	a	a	a	a	r		a
14	r		r	a	a	r	r	a		r
15	r			r	r					r
16	a			r	a					r
17	r			r	r					r
18				a	a					r
19				r	a					
20				a	r					
21				a	a					
22					r					
23					a					
24										

Table 5.16 Action selection for Class 2 in 6-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	a	a
4	a	a	a	r	a	a	a	a	a	a
5	a	r	a	a	a	r	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	r	r	a	a	a	a	a	a	a	a
9	a	r	r	a	a	r	a	r	r	a
10	r	r	a	a	a	a	a	r	r	a
11	a	a	r	r	a	r	a	r	r	r
12	a		r	a	a	r	r	a		a
13	a		r	a	a	a	a	a		a
14	r		a	a	a		a	r		a
15	r			a	a					r
16	r			r	r					r
17	r			a	r					r
18				r	r					r
19				a	a					
20				a	a					
21				a	a					
22				r	a					
23					a					
24					a					

Table 5.17 Action selection for Class 3 in 6-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	r	a	a
3	a	a	a	a	a	a	a	a	a	a
4	a	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	a	a	a	a	a	a	a	a	a
9	a	a	a	a	a	a	a	a	a	a
10	a	a	a	a	a	a	a	a	a	r
11	a	r	a	a	a	r	r	a	a	a
12	a		r	a	a	r	r	a		a
13	r		a	a	a	r	a	a		a
14	a		r	a	a	a	r	a		a
15	r			a	a					a
16	a			a	a					r
17	r			r	a					r
18				a	a					a
19				r	r					a
20				a	r					
21				a	a					
22					r					
23										
24										

Table 5.18 Action selection for Class 4 in 6-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	a	a
4	a	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	a	a	a	a	a	a	a	a	a
9	a	a	a	a	a	a	a	a	r	a
10	a	a	a	a	a	a	a	a	a	a
11	a	a	a	a	a	r	a	a	a	a
12	a		r	a	a	r	r	a		a
13	a		a	a	a	a	a	a		a
14	a		a	a	a	a	a	a		r
15	a			r	a					a
16	a			r	a					a
17	a			a	a					r
18				a	r					a
19				r	a					r
20				a	a					
21				a	r					
22					a					
23										
24										

Table 5.19 Action selection for Class 5 in 6-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	a	a
4	a	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	r	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	a	a	a	a	a	a	a	a	a
9	a	a	a	a	a	a	a	a	a	a
10	a	a	a	a	a	a	a	a	a	a
11	a	r	a	a	a	a	a	a	a	a
12	a		a	a	a	r	a	a		a
13	a		a	a	a	a	r	r		a
14	a		a	a	a		r	a		a
15	a			a	a					a
16	a			a	a					a
17	a			a	a					
18				a	a					
19				r	r					
20				r	r					
21				a	r					
22				r	a					
23					r					
24										

Table 5.20 Action selection for Class 6 in 6-fare-class system

phi	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
1	a	a	a	a	a	a	a	a	a	a
2	a	a	a	a	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a	a	a
4	a	a	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	a	a	a
8	a	a	a	a	a	a	a	a	a	a
9	a	a	a	a	a	a	a	a	a	a
10	a	r	a	a	a	a	a	a	a	a
11	a	a	a	a	a	a	a	a	a	a
12	a		a	a	a	a	a	a		a
13	a		a	a	a	a	a	a		a
14	a			a	a	a		a		a
15	a			a	a					a
16	r			a	a					a
17	r			a	a					
18				a	a					
19				r	r					
20				a	a					
21				r	a					
22					a					
23					a					
24										

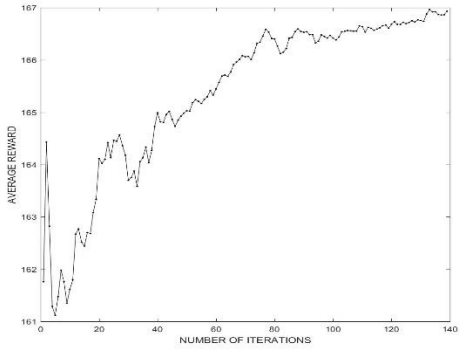

Figure 5.7 Learning phase graph for Case 1 in 4-fare-class system

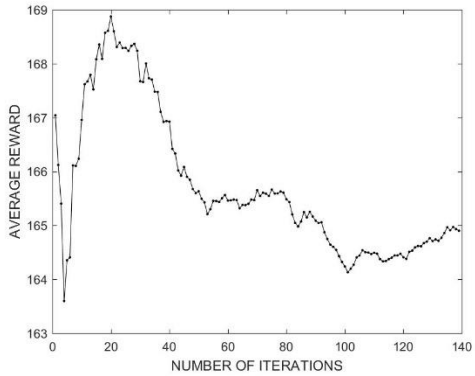

Figure 5.8 Learning phase graph for Case 2 in 4-fare-class system

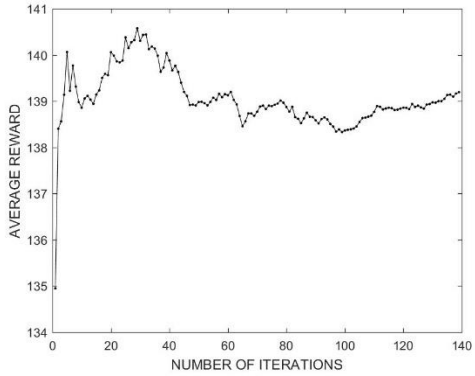

Figure 5.9 Learning phase graph for Case 3 in 4-fare-class system

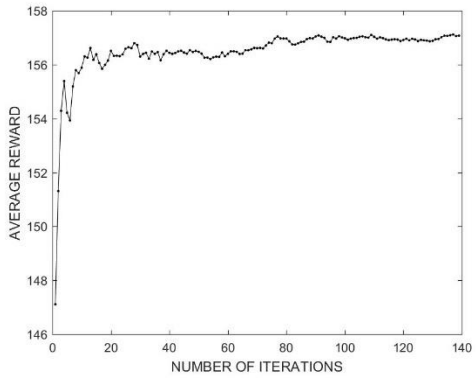

Figure 5.10 Learning phase graph for Case 4 in 4-fare-class system

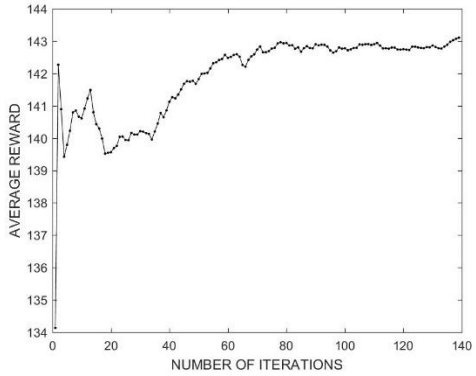

Figure 5.11 Learning phase graph for Case 5 in 4-fare-class system

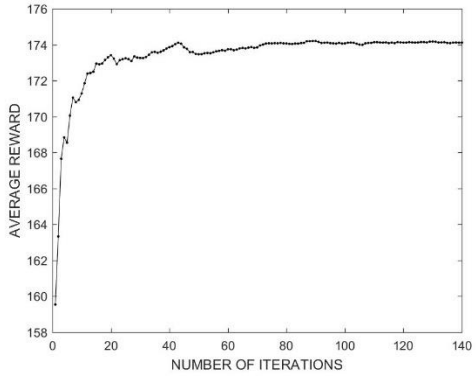

Figure 5.12 Learning phase graph for Case 6 in 4-fare-class system

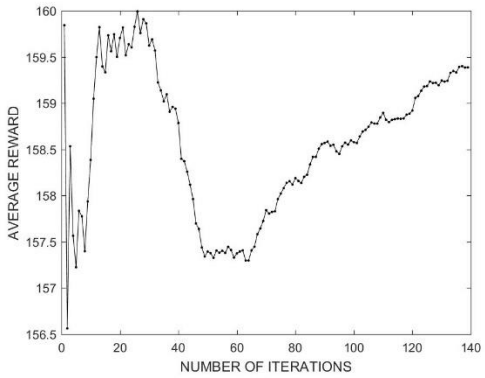

Figure 5.13 Learning phase graph for Case 7 in 4-fare-class system

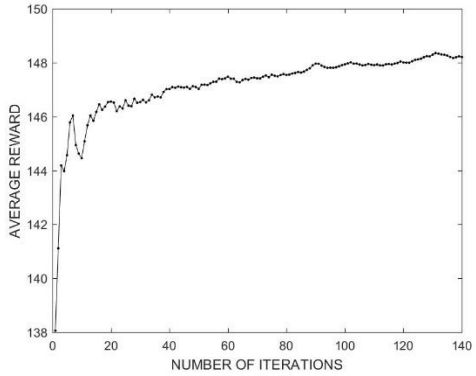

Figure 5.14 Learning phase graph for Case 8 in 4-fare-class system

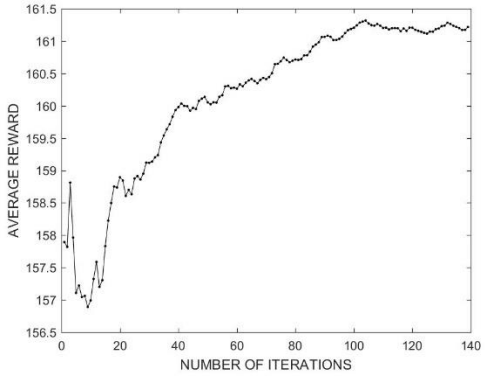

Figure 5.15 Learning phase graph for Case 9 in 4-fare-class system

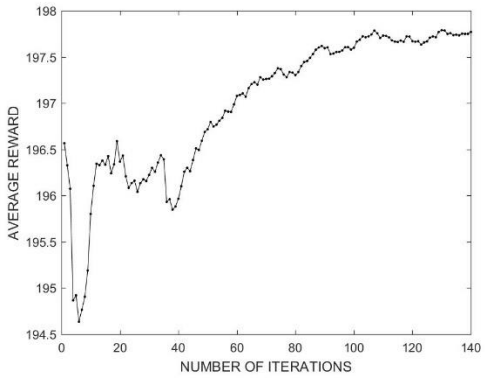

Figure 5.16 Learning phase graph for Case 10 in 4-fare-class system

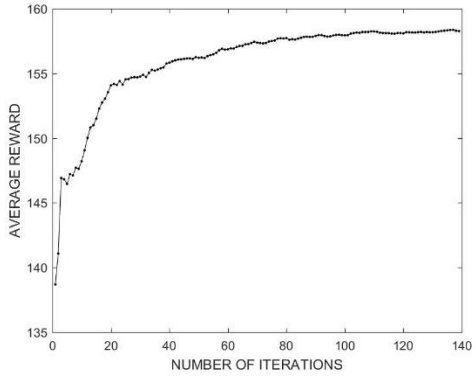

Figure 5.17 Learning phase graph for Case 1 in 6-fare-class system

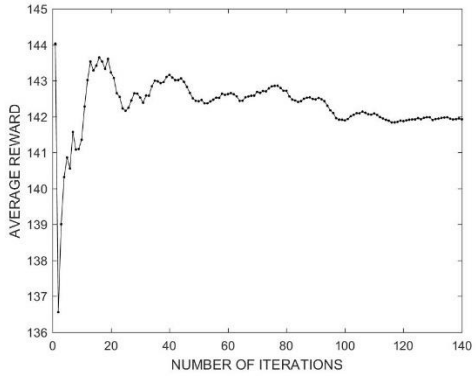

Figure 5.18 Learning phase graph for Case 2 in 6-fare-class system

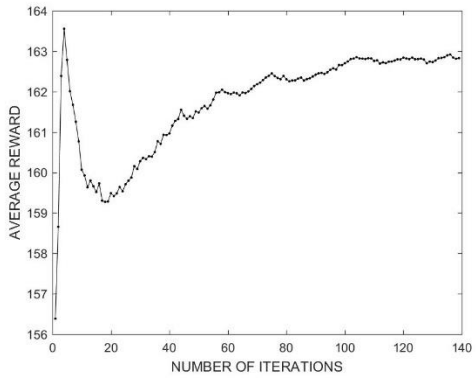

Figure 5.19 Learning phase graph for Case 3 in 6-fare-class system

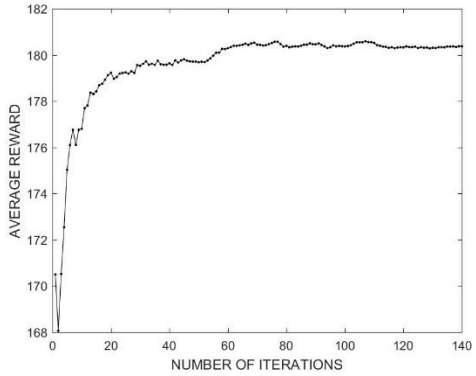

Figure 5.20 Learning phase graph for Case 4 in 6-fare-class system

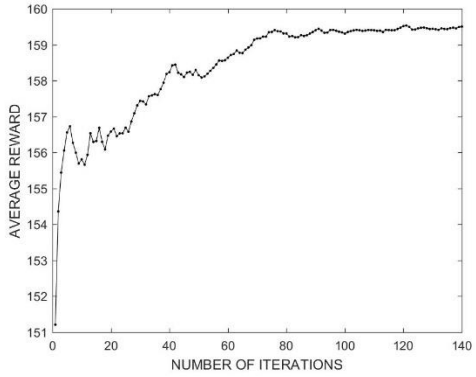

Figure 5.21 Learning phase graph for Case 5 in 6-fare-class system

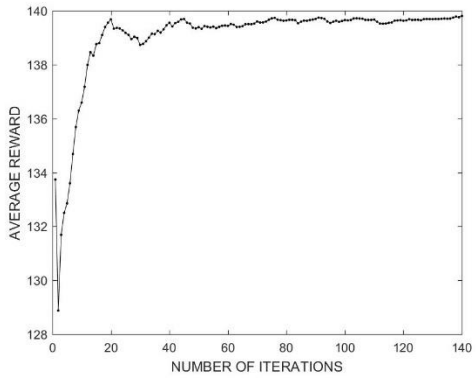

Figure 5.22 Learning phase graph for Case 6 in 6-fare-class system

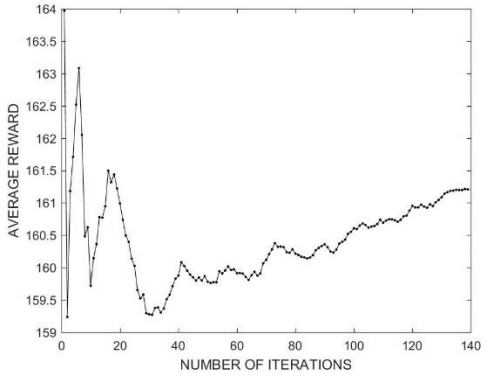

Figure 5.23 Learning phase graph for Case 7 in 6-fare-class system

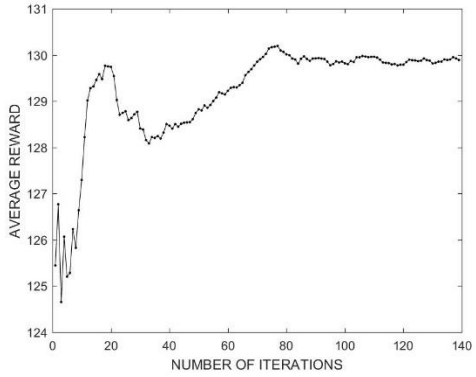

Figure 5.24 Learning phase graph for Case 8 in 6-fare-class system

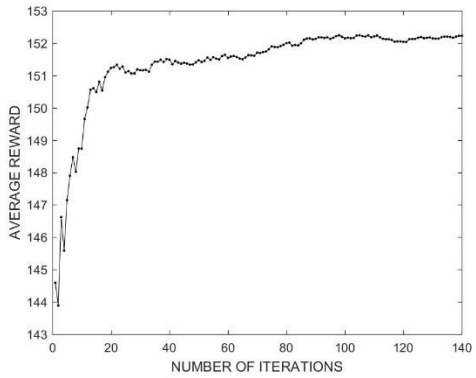

Figure 5.25 Learning phase graph for Case 9 in 6-fare-class system

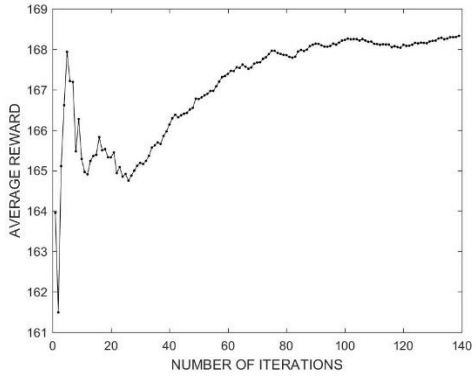

Figure 5.26 Learning phase graph for Case 10 in 6-fare-class system

6. CONCLUSION

This thesis proposed a new version of a reinforcement learning algorithm known as an actor-critic. The new version of the algorithm is designed to overcome a critical difficulty with the traditional actor-critic, i.e., the need to artificially constrain the actor's values. While Gosavi (2014a) proposed an algorithm that constrained the actor's values, the algorithm proposed here improves on that algorithm by developing a new version that provides actor values that have a significantly smaller magnitude.

The algorithm was tested on (i) small MDPs, (ii) small SMDPs, and (iii) two sets of large-scale problems from the airline industry. In the experimentation conducted, the new algorithm was able to outperform the well-known leading airline industry heuristic, namely EMSR-b, which is known to produce excellent results. The improvement in the long-run average reward over EMSR-b ranged from 2.00% to 4.36% for the four fare class systems and from 1.35% to 3.65% for the six fare class systems.

Future work: This research can be extended to the network airline problem that operates within the origin-destination model. Also, the proof of convergence using ordinary differential equations would be another natural extension of this research. The actor-critic algorithm studied in this work can also be applied to other areas of operations engineering, such as queuing, manufacturing, and preventative maintenance.

BIBLIOGRAPHY

Barto A.G., Sutton R.S., and Anderson C.W. (1983) Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Men and Cybernetics,* 13:835-846.

Bellman R. (1957) Dynamic Programming. Princeton University Press, Princeton, NJ.

Belobaba P.P. (1989) Application of a probabilistic decision model to airlines seat inventory control. *Operations Research* 37: 183-197.

Belobaba P. P. (1992) Optimal vs. heuristic methods for nested seat allocation. *ORSA/TIMS Joint National Meeting,* San Francisco, CA.

Bertsekas D.P., Tsitsikikilis J.N. (1996) *Neurodynamic Programming*. Athena Scientific, Belmont, MA

Buffett S. and Scott N. (2004). An algorithm for procurement in supply-chain management. In *Proceedings of the AAMAS 2004 Workshop on Trading Agent Design and Analysis*.

Chaharsooghi S., Heydari J., Zegordi S. (2008) A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems* 45.

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

Durham M. J. (1995) The future of SABRE. *The Handbook of Airline Economics,* D. Jenkins (ed.), pages 485-491.

Gosavi A. (2004) A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning,* 55(1): 5-29.

Gosavi A. (2007) Adaptive critics for airline revenue management. In *Conference Proceedings of the Production and Operations Management Society*, Dallas.

Gosavi A. (2014a) How to Rein in the Volatile Actor: A New Bounded Perspective. *Complex Adaptive Systems,* Pub 4. *Volume 36, pp. 500-507.*

Gosavi A. (2014b) *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning,* Springer, Second Edition.

Gosavi A., Bandla N., Das T.K. (2002) A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking. *IIE Transactions*, 34(9): 729-742.

Gosavi A., Ozkaya E., and Kahraman A. (2007) Simulation optimization for revenue management of airlines with cancellations and overbooking. *OR Spectrum* 29:21-38

Howard R. (1960) *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.

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

Kulkarni K., Gosavi A., Murray S.L., and Grantham K. (2011) Semi-Markov adaptive critic heuristic with application to airline revenue management, *Journal of Control Theory and Applications*, 9(3): 421-430.

Lawhead R., Gosavi A., and Murray S.L. (2017). A bounded actor-critic algorithm for reinforcement learning with applications in airline revenue management, *Intelligent Systems Center Graduate Research Symposium*, Missouri University of Science and Technology. Rolla, Mo.

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),* pages 95-117.

McGill J.I. and van Ryzin G.J. (1999) Revenue management: research overview and prospects. *Transportation Science*, 33(2):233-256.

Pontrandolfo P., Gosavi A., Okogbaa O.G., Das T.K. (2002) Global supply chain management: A reinforcement learning approach. *International Journal of Production Research*, 40: 1299-1317.

Schouten F. V. D. D. and Vanneste S. (1995). Maintenance optimization of a production system with buffer capacity. *European Journal of Operational Research*, 82(2):323–338.

Sennott L. (1999). Stochastic Dynamic Programming and the Control of Queueing Systems. John Wiley and Sons, New York, USA.

Sutton R.S., Barto A.G. (1999) Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA

Talluri K., and Van Ryzin G.J. (2004) *The Theory and Practice of Revenue Management*. Kluwer Academic, Boston, MA.

Talluri K. and van Ryzin G.J. (2005) An introduction to revenue management. *Tutorials in Operations Research, INFORMS.* 142-194.

Venayagamoorthy G., Harley R., and Wunsch D. (2002) Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neuro-control of a turbogenerator. *IEEE Transactions on Neural Networks*, 13 (3):764-773.

Wang Y. and Usher J.M. (2004) Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, 18 (2005): 73-82.

Watkins C. (1989) Learning from delayed rewards, Ph.D. thesis, Kings College, Cambridge, England 1989.

Werbos P.J. (1987) Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man., and Cybernetics*, 17:7-20.

VITA

Ryan Lawhead was born on October 20, 1993 in Arkansas. He received his Bachelor of Science in Petroleum Engineering from Missouri University of Science and Technology in May 2016. Ryan has worked an internship with both Freeport-McMoRan Oil & Gas and L'Oréal USA. In January 2016, he joined the Systems Engineering department at Missouri University of Science and Technology as a dual enrolled student before becoming a full time graduate student in August 2016.

He has presented at his research at the INFORMS Annual Meeting 2016 and 2017. He has also presented a paper at the Intelligent Systems Center Graduate Research Symposium in 2017. His research interests included revenue management techniques, reinforcement learning, and simulation-based optimization techniques. Ryan has been a member of INFORMS. In December 2017, he received his M.S. in Systems Engineering from the Missouri University of Science and Technology, Rolla, Missouri, USA.