2020

DeepARM: An airline revenue management system for dynamic pricing and seat inventory control using deep reinforcement learning

Syed Arbab Mohd Shihab
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

Recommended Citation
https://lib.dr.iastate.edu/etd/18227

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
DeepARM: An airline revenue management system for dynamic pricing and seat inventory control using deep reinforcement learning

by

Syed Arbab Mohd Shihab

A dissertation submitted to the graduate faculty in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY

Major: Aerospace Engineering

Program of Study Committee:
Christina Bloebaum, Co-major Professor
Peng Wei, Co-major Professor
Leifur Leifsson, Co-major Professor
A Ram Kim
Jin Tian

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University
Ames, Iowa
2020

Copyright © Syed Arbab Mohd Shihab, 2020. All rights reserved.
DEDICATION

I would like to wholeheartedly dedicate this thesis to my dear “ammu” (mother) Fatema Yasmin Reba, who will always be the light of my life, my trailblazing “abbu” (father) Syed Anwar Shamim, who is my real life hero and inspiration, my sweet “fuppi” (aunt) Dilruba Chowdhury, who helped me get back on my feet when I was down, and my beloved wife Maisha Tamanna Khan, who is my all-time cheerleader, confidante and partner-in-mischief, and my silver lining in every weather.
# TABLE OF CONTENTS

**LIST OF TABLES** ............................................................................. v

**LIST OF FIGURES** .......................................................................... vi

**ACKNOWLEDGMENTS** .................................................................... xii

**ABSTRACT** .................................................................................. xiv

**CHAPTER 1. INTRODUCTION** .......................................................... 1

1.1 Airline Revenue Management ....................................................... 2

1.2 Traditional Model-Based ARM System ......................................... 8

1.3 Research Motivation .................................................................. 9

1.4 Deep Reinforcement Learning ..................................................... 11

1.4.1 Markov Decision Process .......................................................... 11

1.4.2 Q-learning ........................................................................... 14

1.4.3 Deep Reinforcement Learning: Deep Q-Learning .................... 15

1.5 Applying Deep Reinforcement Learning to Airline Revenue Management ................................................................. 17

1.6 Dissertation Structure ................................................................. 19

**CHAPTER 2. TRADITIONAL AIRLINE REVENUE MANAGEMENT METHODS: LITERATURE REVIEW AND BACKGROUND** .............................................. 20

2.1 Mathematical Programming ....................................................... 21

2.2 Dynamic Programming ............................................................... 24

2.3 Heuristic Rule-Based Decision Model: Expected Marginal Seat Revenue ................................................................. 26

2.4 Summary .................................................................................. 29

**CHAPTER 3. SEAT INVENTORY CONTROL USING DEEP REINFORCEMENT LEARNING** ......................................................................... 30

3.1 Problem Statement .................................................................... 30

3.2 Air Travel Market Simulator ....................................................... 31

3.2.1 Passenger Arrival Model .......................................................... 32

3.2.2 Cancellation Model ................................................................. 34

3.2.3 Optimal Revenue Computation ............................................... 35

3.3 Markov Decision Process Formulation ......................................... 36

3.3.1 States .................................................................................. 36

3.3.2 Actions .................................................................................. 37
CHAPTER 3. DYNAMIC PRICING USING DEEP REINFORCEMENT LEARNING . . 55
  3.1 Problem Statement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
  3.2 Air Travel Market Simulator . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
    3.2.1 Passenger Arrival Model . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
    3.2.2 Cancellation Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
    3.2.3 Optimal Revenue Computation . . . . . . . . . . . . . . . . . . . . . . . . 61
  3.3 Markov Decision Process Formulation . . . . . . . . . . . . . . . . . . . . . . . 63
    3.3.1 States . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
    3.3.2 Actions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
    3.3.3 Model Dynamics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
    3.3.4 Reward Function and Discount Factor . . . . . . . . . . . . . . . . . . . . . 66
  3.4 Solution Method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
    3.4.1 DRL Agent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
    3.4.2 EMSRb Agent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
  3.5 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
    3.5.1 Training in Simulator-T . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
    3.5.2 Testing in Simulator-A . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
    3.5.3 DRL vs Dynamic Programming: Learning the Optimal Dynamic Pricing Policy 76
    3.5.4 Adaptability Testing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
  3.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83

CHAPTER 5. CONCLUSIONS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
  5.1 Key Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
  5.2 Key Research Findings . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
  5.3 Future Research Directions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88

BIBLIOGRAPHY . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.1</td>
<td>An example of fare products offered in the economy cabin of an O-D market</td>
<td>5</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>The fare class mean arrivals per episode in different mean arrival settings</td>
<td>33</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>The beta distribution shape parameter values of the fare classes</td>
<td>34</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>The fare class cancellation rates in different cancellation probability settings</td>
<td>35</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>Actions the agent can take to control the seat inventory</td>
<td>37</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>Performance of the DRL agent in the different numerical experiments</td>
<td>48</td>
</tr>
<tr>
<td>Table 3.6</td>
<td>Performance of the EMSRb agent in the different numerical experiments</td>
<td>48</td>
</tr>
<tr>
<td>Table 3.7</td>
<td>Performance of the DRL agent relative to the EMSRb agent in the different numerical experiments</td>
<td>49</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>The beta distribution shape parameter values of the WTP passenger groups</td>
<td>60</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Actions the agent can take to control the prices of the fare products and the availability of the low fare product</td>
<td>64</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Results of neural network configuration and hyperparameter tuning experiments</td>
<td>69</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Results of different learning algorithms with the best found hyperparameter setting</td>
<td>69</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Results of EMSRb agent’s hyperparameter tuning experiments</td>
<td>72</td>
</tr>
<tr>
<td>Table 4.6</td>
<td>Results of DRL agent’s hyperparameter tuning experiments for adaptability test</td>
<td>80</td>
</tr>
<tr>
<td>Table 4.7</td>
<td>Results of EMSRb agent’s hyperparameter tuning experiments for adaptability test</td>
<td>81</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A single nonstop flight between an O-D pair</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Seat map of an Airbus A330-200 aircraft</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Passenger arrival, booking and cancellation process of different fare products</td>
<td>6</td>
</tr>
<tr>
<td>1.4</td>
<td>Traditional model-based ARM system architecture</td>
<td>9</td>
</tr>
<tr>
<td>1.5</td>
<td>The agent-environment interaction in the RL framework</td>
<td>13</td>
</tr>
<tr>
<td>1.6</td>
<td>Types of seat inventory structure</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Plot of passenger arrivals rates of different fare classes for the second MAS</td>
<td>34</td>
</tr>
<tr>
<td>2.3</td>
<td>The agent-market simulator interaction</td>
<td>41</td>
</tr>
<tr>
<td>3.1</td>
<td>The configuration of the deep neural network used to approximate Q-values</td>
<td>42</td>
</tr>
<tr>
<td>3.2</td>
<td>Average optimal revenue percentage generated by agent during training for MAS = 1 and CPS = 1</td>
<td>44</td>
</tr>
<tr>
<td>3.3</td>
<td>Load factor achieved by agent during training for MAS = 1 and CPS = 1</td>
<td>44</td>
</tr>
<tr>
<td>3.4</td>
<td>Load factor achieved by agent during testing for MAS = 1 and CPS = 1</td>
<td>45</td>
</tr>
<tr>
<td>3.5</td>
<td>Load factor achieved by agent during testing for the ARM MDP with 2 fare classes</td>
<td>45</td>
</tr>
<tr>
<td>3.6</td>
<td>Fare class booking plot of agent during testing for MAS = 1 and CPS = 1</td>
<td>47</td>
</tr>
<tr>
<td>3.7</td>
<td>Optimal revenue percentage generated by agent during training for the ARM MDP with 2 fare classes</td>
<td>47</td>
</tr>
<tr>
<td>3.8</td>
<td>Average optimal revenue percentage generated by DRL agent during training</td>
<td>74</td>
</tr>
<tr>
<td>3.9</td>
<td>Average load factor achieved by DRL agent during training</td>
<td>74</td>
</tr>
<tr>
<td>3.10</td>
<td>Optimal revenue percentage generated by agents in the event of a negative high fare product demand shock (adaptability experiment-1) in simulator-A</td>
<td>81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>The research and performance evaluation methodology for the DR and EMSRb agents employed for dynamic pricing of flights</td>
<td>59</td>
</tr>
<tr>
<td>4.2</td>
<td>Plot of passenger arrivals rates of different WTP passenger groups</td>
<td>61</td>
</tr>
<tr>
<td>4.3</td>
<td>The agent-market simulator interaction</td>
<td>68</td>
</tr>
<tr>
<td>4.4</td>
<td>The configuration of the deep neural network used to approximate Q-values</td>
<td>70</td>
</tr>
<tr>
<td>4.5</td>
<td>Average optimal revenue percentage generated by DRL agent during training</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>Average load factor achieved by DRL agent during training</td>
<td>74</td>
</tr>
<tr>
<td>4.7</td>
<td>Optimal revenue percentage generated by agents during testing in simulator-A</td>
<td>75</td>
</tr>
<tr>
<td>4.8</td>
<td>Load factor achieved by agents during testing in simulator-A</td>
<td>75</td>
</tr>
<tr>
<td>4.9</td>
<td>The DRL agent’s price evolution graph (top) and passenger arrival bar plot (bottom) of test flight episode 166</td>
<td>76</td>
</tr>
<tr>
<td>4.10</td>
<td>Optimal revenue percentage generated by agent during training for the small scale dynamic pricing MDP</td>
<td>79</td>
</tr>
<tr>
<td>4.11</td>
<td>Load factor achieved by agent during training for the small scale dynamic pricing MDP</td>
<td>79</td>
</tr>
<tr>
<td>4.12</td>
<td>Optimal revenue percentage generated by agents in the event of a negative high fare product demand shock (adaptability experiment-1) in simulator-A</td>
<td>81</td>
</tr>
</tbody>
</table>
Figure 4.13  Optimal revenue percentage generated by agents in the event of a surge in high fare product cancellation rates (adaptability experiment-2) in simulator-A
### Nomenclature

- **$A$** The action space of a MDP
- **$S$** The state space of a MDP
- **$a$** An action in a MDP
- **$s, s'$** A state in a MDP
- **$t$** Time step
- **$M$** Batch size or number of experience samples in the mini-batch
- **$Q^*(s_t, a)$** Optimal state-action value or Q-value of a state $s_t$ and action $a$
- **$T$** The time horizon of a MDP
- **$U^*(s_t)$** Optimal value of a state $s_t$
- **$\alpha$** Learning rate or step size
- **$\mathcal{B}(s, a)$** Set of basis functions given as input to a perceptron
- **$\epsilon$** Probability of taking a random action
- **$\gamma$** Discount factor
- **$\pi^*$** Optimal policy
- **$\theta$** Neural network weight vector
- **$a^*$** Optimal action
- **$m$** Experience sample index
- **$r_t$** Reward received at time step $t$
- **$s_t$** The state of the agent at time step $t$
- **$L$** Total number of flight legs in the airline network
- **$l$** Flight leg index
odf  Origin-destination-fare class index
$E[D_{odf}]$  Expected demand associated with fare class $odf$
$f_{odf}$  Fare associated with fare class $odf$
$x_{odf}$  Number of allocated seats associated with fare class $odf$
$S_l$  Set of fare classes associated with leg $l$
$\kappa$  Flight capacity
$\kappa_l$  Flight capacity of leg $l$
$I$  Set of all fare classes
$i$  Fare class index
$BL_i$  Booking limit of a fare class $i$
$D_i$  Mean demand of fare class $i$
$P(d_i > s_i)$  Probability of $d_i > s_i$
$\bar{\kappa}$  Authorized flight capacity
$\bar{f}_i$  Demand-weighted average of revenue values of fare class $i$ and all higher fare classes
$\rho_i$  Optimal number of seats to protect for fare classes 1 through $i$ from the fare classes $i+1$
$d_i$  Demand of fare class $i$
$f_i$  Revenue value of fare class $i$
$\text{EMSR}(s_i)$  EMRS from protecting the $s$-th seat of fare class $i$
$\lambda_i(t)$  Mean arrival rate of fare class $i$ at time step $t$
$\tilde{\alpha}_i, \tilde{\beta}_i$  Beta distribution with shape parameters
$A_{t,i}$  Number of arrivals of fare class $i$ at time step $t$
$\Gamma$  Gamma function
$\lambda'_i(\tau; \tilde{\alpha}_i, \tilde{\beta}_i)$  Probability density function value of a beta distribution with shape parameters $\tilde{\alpha}_i$ and $\tilde{\beta}_i$ corresponding to fare class $i$ at time $\tau$
$\hat{\lambda}_i$  Mean number of arrivals of fare class $i$ in each flight episode
$A_{t,e}^T$  Total number of passenger arrivals of fare class $i$ in episode $e$
\( C_{i,e}^{T} \) Total number of passengers of fare class \( i \) who will cancel their booking at some point before flight departure in episode \( e \)

\( \tilde{k}_{i,e} \) Remaining flight capacity after the optimal number of passengers from fare classes higher than fare class \( i \) has been accommodated

\( e \) Flight episode index

\( n_{i,e}^{*} \) Optimal number of passengers to book from fare class \( i \) in flight episode \( e \)

\( C_{t,i} \) Total number of booking cancellations of fare class \( i \) at time step \( t \)

\( \alpha_{t,i} \) Number of passenger arrivals of fare class \( i \) in time step \( t \)

\( \beta_{t,i} \) Total number of new bookings of fare class \( i \) at time step \( t \)

\( b_{i} \) Number of bookings of fare class \( i \)

\( B \) Bumping cost factor

\( \eta^{T} \) Total number of passengers that need to be bumped

\( \eta_{i} \) Number of passengers bumped from fare class \( i \)

\( \mu_{LF} \) Average load factor

\( \mu_{RP^*} \) Average percentage of optimal revenue generated

\( \mu_{R} \) Average revenue generated

\( \sigma_{LF} \) Standard deviation of the load factor

\( \sigma_{RP^*} \) Standard deviation of the percentage of optimal revenue generated

\( \sigma_{R} \) Standard deviation of the revenue generated

\( \text{CI} \) Confidence interval

\( P(\Delta b_{t,i}) \) Probability of fare class \( i \) bookings changing by \( \Delta b_{t,i} \) at time step \( t \)

\( J \) Set of all fare products

\( j \) Fare product index

\( k \) WTP index

\( l \) Price point index

\( A_{t,k} \) Number of arrivals of WTP \( k \) passenger group at each time step \( t \)

\( \eta_{l} \) Number of passengers bumped at the end of the booking period who paid fare \( l \)
\( \lambda'_k(\tau; \hat{\alpha}_k, \hat{\beta}_k) \) Probability density function value of a beta distribution with shape parameters \( \hat{\alpha}_k \) and \( \hat{\beta}_k \) corresponding to passenger group with WTP \( k \) at time \( \tau \)

\( \lambda_k(t) \) Mean arrival rate of passenger group with WTP \( k \) at time step \( t \)

\( \hat{A}_k \) Mean number of arrivals of passenger group with WTP \( k \) in each flight episode

\( n^B \) Total number of passengers booked

\( n'_k \) Number of arrivals of passenger with WTP \( k \) at time step \( t \)

\( p^l_t \) A binary decision variable indicating whether price point \( S/l \) is selected or not for its corresponding fare product at time step \( t \)

\( C_{t,l} \) Total number of booking cancellations associated with price point \( l \) at time step \( t \)

\( \alpha_{t,l} \) Number of passenger arrivals with WTP \( l \) in time step \( t \)

\( \beta_{t,l} \) Total number of new bookings associated with price point \( l \) at time step \( t \)
ACKNOWLEDGMENTS

I am indebted to my research advisors Dr. Christina Bloebaum and Dr. Peng Wei for their ceaseless and wholehearted mentorship and support in many different forms throughout my graduate academic journey. They have helped me gain a deeper understanding of optimization, decision-making, reinforcement learning and airline operations planning, which are the primary areas of study that form the foundation of this research work. At times, when my research met a seemingly inescapable dead end or a major technical hurdle, they came to my rescue by encouraging me to explore new directions and think out of the box. On weekdays and weekends, during day and night, they were always an email or message away, eager to lend an ear and provide consultation whenever it was necessary. Their confidence in my potential and that of this work has motivated me to transcend my limits, and publish this research and present it at various conferences. Beyond assisting me in my research, they have also helped me strengthen my presentation, communication and teaching skills by giving me numerous opportunities to practice these skills and providing constructive feedback to help me prepare for these opportunities.

I am also very thankful to my Ph.D. committee for their invaluable recommendations that have shaped the research methodology followed in this research work and greatly improved its rigor. Many insightful results generated in this work can be attributed to their recommendations. I am also especially grateful to Dr. Leifur Leifsson for meticulously reviewing this dissertation line-by-line cover-to-cover. His feedback has significantly refined the overall write-up, contents, format and structure of this document.

My lab mates, department staff, friends and neighbors also deserve a special and loud shout-out here for making the lab and the city a fun, lively and productive place to work and live at. I am grateful to Nazareen Sikkandar-Basha for her friendly company, counsel and assistance while carrying out this research. I have learned many document editing and design hacks from her, which has saved me
a great deal of time and made my work more presentable. I am also grateful to Marc Brittain, Xuxi Yang and Josh Bertram for generously sharing their programming and AI knowledge and expertise with me and helping me troubleshoot technical issues during the implementation of the DeepARM system. Without their assistance, I would have had to spend hours on question and answer sites, like Stack Exchange, Quora and GitHub, to find the solutions to my problems. On a similar note, I would also like to express my gratitude to Marisa Mendoza and Paul Easker for their exceptional administrative and IT technical support respectively, which has greatly facilitated the execution of this research. I am also thankful to Tajbir Ahmmed Khan, Monira Parvin Mini and Munaf Asbat Ahmmed for regularly checking on me to see if I needed anything and keeping me nourished with homemade Michelin 3-Star quality food during the most critical and stressful period of this academic journey.

Finally, I would like to thank all my family members who pushed me to pursue my dreams and persevere in the face of seemingly insurmountable challenges. My niece Syeda Shaheerah Sanaya was never failed to cheer me up and give me an escape from all my stress and worries with her charming and effervescent personality, and thus saving me from many burnouts. Like Sanaya, her mother Shagufta Shahzeen Siddiqui, who is my “bhabi” (sister-in-law), reminded me in her own way to keep breathing, living and enjoying life amidst all the hustle and bustle of daily life through our interesting conversations. During times of uncertainty and unfavorable circumstances, my brother Syed Shahrear Shabab kept me moving forward with his high impact, penetrating pep talks. My “mama” (maternal uncle) Prof. AKM Ibrahim Chowdhury Ratan, “chachu” (paternal uncle) Syed Sarfaraz Shah, “maa” (mother-in-Law) Tahmina Hossain and “nana bhai” (grandfather) Syed Iqbal Hossain always stayed in touch with me to ensure I was safe, healthy and productive, and rooted for me throughout this journey. Last but not least, neither this dissertation would have been possible and nor would I have been able to come to this point in my career without the unconditional and selfless love, support, prayers and sacrifices of the people to whom this dissertation is dedicated to.
ABSTRACT

Commercial airlines use revenue management systems to maximize their revenue by making real-time decisions on the prices and booking limits of different fare products and classes offered in each of its scheduled flights. Traditional approaches — such as mathematical programming, dynamic programming and heuristic rule-based decision models — heavily rely on mathematical models of demand and passenger arrival, choice and cancellation, making their performance sensitive to the accuracy of these model estimates. Moreover, many of these approaches scale poorly with increase in problem dimensionality. Additionally, they lack the ability to explore and “directly” learn the true market dynamics from interactions with passengers and adapt to changes in market conditions on their own. To overcome these limitations, this research uses deep reinforcement learning (DRL), a model-free decision-making framework, for finding the optimal policies of seat inventory control and dynamic pricing problems. The DRL framework employs a deep neural network to approximate the expected optimal revenues for all possible state-action combinations, allowing it to handle the large state spaces of the problems. Multiple fare classes with stochastic demand, passenger arrivals and booking cancellations, and overbooking have been considered in the problems. An air travel market simulator was developed based on the market dynamics and passenger behavior for training and testing the agent. The results demonstrate that the DRL agent is capable of learning the optimal airline revenue management policy through interactions with the market, matching the performance of exact dynamic programming methods. The performance of the agent in different simulated market scenarios was found to be close to the theoretical optimal revenues and superior to that of the expected marginal seat revenue-b (EMSRb) method. Also, when faced with market perturbations, the DRL agent has been observed to actively learn to change its policy to maximize revenue in the new environment, demonstrating its ability to adapt to changes in the market conditions.
CHAPTER 1. INTRODUCTION

Few markets are as fiercely competitive as the current air travel market. This intense competition dates back to the deregulation of the airline industry in 1978, after which US airlines could freely set up their route network and quote fares for their offered itineraries. Since then, airline corporations have been heavily relying on airline revenue management (ARM) systems, a decision support tool designed for maximizing the total expected revenue generated from the sale of tickets in all their flights [Belobaba et al. (2015)]. These systems strive to achieve their goal by optimally setting the prices and booking limits of the different types of tickets offered by the airlines in each flight at each decision-making instance based on the time remaining till flight departure, unused flight capacity (seat capacity), demand and passenger characteristics. Considering the large scale nature of operations of traditional network airlines, such as Delta Airlines, even small revenue increments per flight provided by ARM is significant: the sale of only one seat per flight at full price instead of the discount rate can contribute an additional $50 million to the airline’s annual revenues [Cross (2011)]. In today’s highly competitive airline industry, even if an airline carries out its fleet planning, route planning, scheduling and other airline operations planning optimally, it still may not perform well financially if it does not optimally conduct its revenue management.

In the remainder of this introductory chapter, the problem of ARM in the context of airline business practices is detailed in the first section. The architecture of the traditional model-based system widely used in practice to solve the ARM problem is described in the next section. The research motivation for this work, the key concepts of our proposed model-free decision-making framework for ARM, the DRL framework, and the application of DRL to ARM are then discussed successively in the following three sections. An overview of the key contributions of this research
Figure 1.1: A single nonstop flight between an O-D pair

is given in the following section. Finally, this chapter concludes with an outline of the rest of the dissertation.

1.1 Airline Revenue Management

The characteristics of the ARM problem is defined by the airline business model, which may vary in some respects for different types of airlines such as legacy network carriers, low cost carriers and ultra low cost carriers. All passenger airlines operate scheduled flights between different origin-destination (O-D) pairs in their route network. Figure 1.1 illustrates the operation of a nonstop flight between one such O-D pair. For each of these flights, they have a limited number of seats to sell in a limited amount of time as specified by the flight capacity of the aircraft used by the airline and the booking period (or sales horizon) for that flight respectively. The constraint of flight capacity is visualized in Fig. 1.2 [The Flight (2019)], which shows the seat map of the economy, business and first class cabins of an Airbus A330-200 aircraft.

Passengers with different characteristics, such as willingness-to-pay (WTP), schedule flexibility and purchase commitment ability, arrive in the airline booking system at different times throughout the booking period to book a seat (buy a ticket). In the context of air travel, the WTP (also known as the personal reservation price) of a passenger is the maximum fare the passenger is willing to pay for booking a ticket. Based on these characteristics, passengers can be broadly categorized into two groups: business and leisure (or vacationers). Business passengers are generally more price inelastic than leisure travelers.
In theory, the revenue generated from a flight is maximized when its seats are sold to passengers at a price equal to their WTP in descending order. In other words, in any given flight, if the airline can charge each passenger their WTP and there are $n$ seats, then it should fill its aircraft to capacity with the $n$ highest WTP passengers seeking to book that flight to generate the maximum possible revenue. In practice, it is difficult for airlines to obtain this theoretical maximum revenue due to several reasons. Firstly, airlines cannot determine the WTP of each passenger arriving in their booking system with certainty. Secondly, they do not know the exact number of passenger arrivals for the flight in advance during the booking period. Thirdly, passengers with higher WTP typically arrive later than ones with lower WTP. Lastly, passengers tend to buy the cheapest tickets that fulfill their requirements, making it difficult for airlines to collect fares equal to the WTP of the passengers. Clearly, the policy of accommodating passengers in a first-come-first-serve basis is not an optimal one as the airline may not have enough seats left for all high WTP passengers near the end of the booking period for flights where the total number of passenger arrivals is greater than the seat capacity. As a result, the airline may miss out on revenue from not being able to collect higher fares from the high WTP passengers.

To deal with these challenges, each airline uses an ARM system to maximize revenue by pricing and enforcing booking limits to control seat inventory based on estimates of demand, passenger arrival times and WTP, and booking cancellation and no-show rates.

The practice of ARM begins with product differentiation, a process of creating various *fare products* for each unique O-D itinerary through different combinations of service amenities and fare
restrictions (or fare rules) to take advantage of the differences in passenger characteristics. Examples of restrictions include advance purchase of ticket, a Saturday night minimum stay or a min/max stay for a certain number of days, change fees, cancellation fees, etc. The service amenities of a fare product determines the number of allowed luggage, the allowed size for each luggage, eligibility for food and beverages, seat selection, priority boarding, earned miles, etc. An example of some typical fare products offered within the economy cabin by an airline and the set of restrictions associated with them are shown in Table 1.1. The group of fare products offered in each O-D market by an airline is collectively called a fare structure. These fare products are created to segment the O-D markets so that passengers with high WTP do not find lower priced products appealing and hence do not “buy down”. As a result, passengers generally end up purchasing products with prices close or equal to their WTP. For more details on the practice of using sale restrictions to segment demand, the reader is referred to [Li (2001)].

Once the fare products are defined, the next step in the ARM process is pricing the fare products, which involves determining a set of finite number of price points for each fare product based on the airline’s business strategy, competitive environment of each O-D market in the network, and other factors. The limitation of having to choose a price for any particular fare product at any given time during the booking period from its corresponding finite set of price points as opposed to any arbitrary price is a result of the legacy airline ticket distribution standards and technological infrastructure currently used by the online travel agents, brick and mortar retailers, corporate travel services, and metasearch websites; these standards and technologies were originally developed prior to the creation of the internet [Niketic and Mules (1993); Vinod (2015)]. To overcome this limitation, the International Air Transport Association proposed an advanced distribution standard called the New Distribution Capability (NDC). With the expected widespread adoption of NDC in the near future, airlines would soon not only be able to arbitrarily price their fare products dynamically in real time, a practice known as continuous pricing in the airline industry, but also create customized offers for individual passengers, a practice known as personalized pricing in the airline industry [Hoyles (2015); Durivaux (2018)]. The association of a fare product belonging to a certain O-D itinerary
(along with its set of restrictions) with a price point leads to the creation of a fare class or booking class. These fare classes are then typically filed (or published) with a central distribution agency such as the Airline Tariff Publishing Company (ATPCO) [Vinod (2010)]. At any given time, at most one fare class corresponding to each O-D fare product is open for sale. In the example given in Table 1.1, there are two fare classes associated with Economy Basic, two with Economy Flexible, and three with Economy Premium.

The typical arrival process of passengers seeking to book different fare products during a booking period of 365 days is depicted in Fig. 1.3. Here, the light green, green, orange and blue colored passengers are looking to buy Economy Basic, Economy Flexible, Business class, and First class fare products respectively. Passengers interested in lower (cheaper) fare products (and fare classes) typically arrive earlier than the ones of more higher (expensive) fare products (and fare classes). Airlines generate revenue from the sale of tickets of different fare products. One of the Economy Flexible passenger is shown to cancel the booking shortly after he/she has purchased it. In the case of such cancellations, the airline loses some of the collected revenue as it provides fare reimbursements to these passengers, which is equal to the amount remaining after cancellations fees are deducted from the fares paid by the passengers.

The booking period, typically spanning for a duration between one year to six months, may be divided into several small time intervals or steps during which the available fare classes remain unchanged. The beginning of each of these time steps can be considered to be a decision-making instance when the ARM system computes the booking limits — the number of available seats — for all the fare classes based on the distributions of demand, passenger inter-arrival times, cancellations and no-show rates of each fare class, and the remaining time to departure and seat capacity. These

<table>
<thead>
<tr>
<th>Economy Basic</th>
<th>Economy Flexible</th>
<th>Economy Premium</th>
</tr>
</thead>
<tbody>
<tr>
<td>No cancellations allowed</td>
<td>Cancellations allowed for a fee</td>
<td>Free cancellations</td>
</tr>
<tr>
<td>No changes allowed</td>
<td>Changes allowed for a fee</td>
<td>Free changes</td>
</tr>
<tr>
<td>Min/max stay required</td>
<td>No min/max stay requirements</td>
<td>No min/max stay requirements</td>
</tr>
<tr>
<td>Fare: ${100, 125}$</td>
<td>Fare: ${200, 300}$</td>
<td>Fare: ${400, 600, 700}$</td>
</tr>
</tbody>
</table>
decision-making instances are also called data collection points (DCPs) as this is when airlines collect data on the latest bookings in a given flight.

When the booking limit of a fare class is reached, the ARM system closes the fare class to make it unavailable for sale. It is necessary to set booking limits for the lower fare classes to protect seats for later arriving high WTP passengers. By opening and closing different fare classes associated with a fare product, the ARM system is essentially varying the price of the fare product. Because passengers arriving close to departure generally have higher WTP than the earlier arriving ones, the ARM system typically increases the prices of the fare products by opening higher fare classes and closing lower fare classes as the departure date approaches, a practice known as temporal price discrimination. As the ARM system periodically recomputes the booking limits, it can dynamically adjust the prices of the fare products based on stochastic demand in every flight.

The availability of fare classes is controlled at the flight leg-level in leg-based ARM systems and at the network-level in network-based ARM systems (also known as O-D control systems). As their names imply, network-based ARM systems aim to maximize total flight revenues across the network while leg-based ARM systems strive to maximize flight revenues at each flight leg of the network separately. While network-based systems have been found to produce 1-2% higher network revenues than their leg-based counterparts [Williamson (1992) and Belobaba (2002)], they have substantially higher implementation costs. As there are typically many O-D markets within an airline’s network, the total number of fare classes across all flight legs in the network may be tremendously high. So, for the sake of computational tractability in network ARM, all of these fare classes are typically
mapped to a smaller number of groups, known as *revenue value buckets* or *virtual inventory/fare classes*, in each flight leg of a given fare class’s O-D itinerary based on their “network revenue value”, a method pioneered by American Airlines [Smith and Penn (1988)]. The network revenue value of a fare class is computed by subtracting its *displacement cost* from its price. The displacement cost of a multi-leg fare class passenger is the estimated cost incurred due to the displacement of passengers and revenue on down-line and/or up-line legs of the fare class itinerary. The resulting set of fare classes in each leg and their network revenue values is collectively called a displacement adjusted virtual nesting (DAVN) inventory structure. Without the adjustment of prices of fare classes by their displacement costs, network-based ARM systems tend to give preference to long-haul connecting passengers over short-haul “local” single-leg passengers. Studies have found that the use of DAVN method produces higher network revenues by overcoming this bias towards long-haul connecting passengers [Hornick (1993), Wei (1997), Hung (1998)].

Henceforth, the problem of dynamically determining the optimal booking limits of each (virtual) fare class for maximizing flight revenues is referred to as the *seat inventory control* problem, which is also known as the fare class mix optimization problem. On the other hand, the problem of dynamically selecting the optimal prices from the set of pre-filed price points for each fare product to maximize flight revenues is referred to as the *dynamic pricing* problem in this dissertation. The seat inventory control approach and the dynamic pricing approach are essentially two different approaches to the problem of ARM which share the same objective. In the former approach, the control variables are both price and booking limits whereas in the latter it is only price.

The seat capacity of a flight specified in the ARM system is sometimes more than the actual value. This allows airlines to overbook their flights to avoid missing out on revenue from flying empty seats due to certain passengers cancelling their booking or not showing up. Ideally, the *load factor* of a flight — the ratio of the total number of passenger bookings to the actual flight capacity at flight departure — should be one after accounting for all cancellations and no-shows. If the overbooking is carried out too conservatively, the airline may still miss out on some revenue from flying empty seats. On the other hand, if the overbooking is done too aggressively, the airline
may face the undesirable situation where the number of passengers showing up for the flight at the boarding gate is more than the flight capacity, leaving the airline with no option but to deny boarding to or “bump” a few passengers to bring the load factor down to one. When this happens, the airline needs to adequately compensate the displaced passenger(s) by accommodating them in another flight and/or giving them monetary compensation for their inconvenience and arrival delay at their destination. The cost associated with bumping a passenger is known as the *bumping cost*.

### 1.2 Traditional Model-Based ARM System

The architecture of a traditional model-based ARM system is depicted in Fig. 1.4. As discussed in the preceding section, such systems require the distributions of demand, passenger inter-arrival times, cancellations, and no-shows of each fare class as inputs. These distributions are encoded in mathematical models of demand and passenger arrival, choice and cancellation behavior for each fare class. Other inputs typically include the flight capacity, fare product characteristics, historical booking data, cancellation and no-show data, and current booking data. The mathematical models are estimated from the historical data. All the inputs go to the ARM engine, inside which there are primarily two components, a forecaster and an optimizer (or solver). Given these inputs, the forecaster estimates the fare class demands for the flights. Using the forecasts and other inputs, the optimizer then determines the optimal booking limits for each fare class to maximize the flight revenues subject to the constraint of the remaining flight capacities. The three types of solution methods commonly used to build the optimizer are mathematical programming, dynamic programming and heuristics or decision-rules. The different types of problem formulations and solution methods are detailed in Chapter 2. When a passenger arrives to make a booking, they see the fares of the open fare classes and their seat availability and make a decision on the basis of this information. Whenever a passenger buys or cancels a ticket of a certain fare class, the booking or cancellation information is fed back to the database to update the data. A similar data update process also occurs for no-shows at the time of flight departure.
1.3 Research Motivation

Although traditional model-based ARM systems have helped airlines achieve incremental revenues of varying amounts, they inherently suffer from the following drawbacks:

1. As these systems require models of market dynamics and passenger behavior as inputs to carry out their computations, their performance is only as good as the accuracy of these models, which is inevitably limited due to several factors. Firstly, no single airline or ARM system developer has access to all the relevant historical market and passenger behavior data. Secondly, historical data does not accurately capture all the relevant information on the explanatory variables needed to estimate the models such as passenger characteristics (schedule preference, income, WTP, choice set, buy-up/buy-down tendencies, demographics, etc.), unconstrained demand, demand spill, etc. Thirdly, historical data does not guarantee to be representative of the future. Lastly, market conditions and passenger behavior change with time.

2. Some of the traditional solution methods commonly employed by these systems, namely, mathematical programming and dynamic programming, scale poorly with increase in problem
dimensionality. For any given ARM problem formulation, the number of decision variables associated with each flight leg or O-D itinerary in the problem increases with increase in the number of fare products, price points and decision-making instances. As the scale of the problem increases, the computation run time of these methods become prohibitively high.

3. These systems do not have any room for (online) exploration to try out different ARM strategies to learn the true market dynamics and passenger characteristics. In other words, there is no scope to deviate from the supposedly “optimal” ARM seat inventory control and pricing policies for experimenting with other policies to determine the true global optimal policy.

4. These systems are not capable of “actively” and “directly” learning changes in market dynamics and passenger characteristics by themselves from their interactions with passengers. Unless the models are updated to reflect the changes, these systems assume the market to be stationary and keep following a suboptimal policy.

The goal of this research is to overcome these shortcomings of traditional model-based ARM systems. To do so, this research proposes using DRL, a model-free decision-making framework capable of directly learning the optimal policy from interactions with the market. The DRL framework permits exploration to find the true optimal policy and handles high problem dimensionality or large state-action spaces by leveraging a deep neural network to approximate the expected return (or payoff) from following the optimal policy. To investigate the potential advantages of using this method for ARM, a DRL-based ARM system was developed to learn the optimal seat inventory control and dynamic pricing policies. Henceforth, this system is referred to as DeepARM in this dissertation. Additionally, an air travel market simulator was built to train and test the performance of DeepARM relative to traditional ARM methods. The specific research questions investigated in this work are:

1. a) Can DeepARM learn the optimal policy for controlling seat inventory of flights with multiple fare classes characterized by stochastic demand, passenger arrivals and booking cancellations to maximize flight revenues?
b) Can DeepARM perform better than the traditional EMSRb method in the seat inventory control problem?

2. a) Can DeepARM learn the optimal policy for dynamic pricing of flights with multiple fare products characterized by stochastic demand, passenger arrivals, passenger WTP and booking cancellations to maximize flight revenues?

b) Can DeepARM perform better than the traditional EMSRb method in the dynamic pricing problem?

3. Can DeepARM autonomously adapt to changes in market conditions and passenger behavior?

1.4 Deep Reinforcement Learning

In reinforcement learning (RL), the key idea is to let an artificial intelligence (AI) agent learn the optimal policy based on the rewards it receives from interactions with the environment. An optimal policy is generally defined to be the policy that maximizes the total expected reward. It has been successfully applied for solving sequential decision-making problems in many fields, such as games, robotics, natural language processing, computer vision, neural architecture design, business management, finance, healthcare, Industry 4.0, smart grid, intelligent transportation systems, and computer systems [Li (2017)].

1.4.1 Markov Decision Process

The first step in RL involves formulating the problem as a MDP. A MDP [Bellman (1957)] is generally composed of six components: 1) a set of states $s$ in the environment referred to as the state space $S (s \in S)$; 2) a set of actions $a$ referred to as the action space $A (a \in A)$; 3) a reward function $R(s, a, s')$ which specifies the reward the agent receives from the environment when it moves from state $s$ to state $s'$ by taking action $a$; 4) a transition function $T(s, a, s')$ which specifies the probability of the agent moving from state $s$ to state $s'$ if it takes an action $a$ in state $s$; 5) a discount factor $\gamma$ which specifies the worth of present rewards relative to future rewards ($\gamma \in [0, 1]$) to the agent; and
6) the initial state of the agent. A time step (or time index) or step count $t$ is used to keep track of the time or the number of actions taken by the agent. The total number of time steps or actions (decisions) involved in one episode of the problem determines the problem horizon $T$. At any given time step $t$, the state of the agent is denoted by $s_t$. Note that the state $s_t$ may be any state $s$ within $S$. At any $t \leq T$, the agent gets to take a valid action $a$, which causes it to probabilistically move from from $s_t$ to $s_{t+1}$ based on $T(s_t, a, s_{t+1})$ and receive a reward $r_t = R(s_t, a, s_{t+1})$, as shown in Fig. 1.5. This process repeats until $t = T$ and the agent is at $s_T$, from where any action taken by the agent leads to the termination of the episode. The goal of solving the MDP is to find an optimal policy $\pi^*: s_t \mapsto a^*$, which maps each state $s_t$ to an optimal action $a^*$.

The optimal policy $\pi^*$ for a MDP is generally defined to be the policy that maximizes the total cumulative expected reward [Kochenderfer (2015)]:

$$\pi^* = \arg\max_\pi \mathbb{E} \left[ \sum_{t=0}^T \gamma^t R(s_t, a_t, s_{t+1}) | \pi \right], \quad (1.1)$$

where $a = \pi(s_t)$. The optimal value of a state $s_t$ ($U^*(s_t)$), is defined as total expected reward obtained from state $s_t$ by acting optimally, and is given by the Bellman equation:

$$U^*(s_t) = \max_{a} \sum_{s_{t+1} \in S} T(s_t, a, s_{t+1}) (R(s_t, a, s_{t+1}) + \gamma U^*(s_{t+1})). \quad (1.2)$$

The equation above is called the value function for the optimal policy. In the value function, $\sum_{s_{t+1} \in S} T(s_t, a, s_{t+1}) R(s_t, a, s_{t+1})$ is the expected immediate (current time step) reward at state $s_t$ and $\sum_{s_{t+1} \in S} T(s_t, a, s_{t+1}) \gamma U^*(s_{t+1})$ is the expected discounted future reward at state $s_t$. The optimal state-action value or $Q$-value of a state $s_t$ and action $a$ ($Q^*(s_t, a)$) is defined as the total expected reward obtained from state $s_t$ by first taking action $a$ and then acting optimally afterwards:

$$Q^*(s_t, a) = \sum_{s_{t+1} \in S} T(s_t, a, s_{t+1}) (R(s_t, a, s_{t+1}) + \gamma \max_a Q^*(s_{t+1}, a')) \quad (1.3)$$
The equation above is called the Q-function or optimal action-value function. Note that the expression for the expected immediate reward is the same as before, but the expression for the expected discounted future reward at state $s_t$ is now $\sum_{s_{t+1} \in S} T(s_t, a, s_{t+1}) \gamma U^*(s_{t+1})$, which is computed using Q-values instead of $U^*(s)$.

The optimal action $a^*$ at a state $s_t$ ($\pi^*(s_t)$) is the action that gives the highest expected return. If the transition function, reward function and values of states are known, then the optimal policy (the optimal action at each state), can be computed using:

$$\pi^*(s_t) = \arg\max_a \sum_{s_{t+1} \in S} T(s_t, a, s_{t+1})(R(s_t, a, s_{t+1}) + \gamma U^*(s_{t+1})).$$  

(1.4)

If all the Q-values are known, then the optimal policy can be computed using

$$\pi^*(s_t) = \arg\max_a Q^*(s_t, a),$$

(1.5)

in which case there is no need to explicitly know the reward and transition functions.
1.4.2 Q-learning

Q-Learning [Watkins and Dayan (1992)] is a popular model-free RL algorithm for determining the Q-values when the reward and transition functions of a MDP are unknown. The algorithm estimates the Q-values based on the rewards the agent receives through interactions with the environment. It iteratively updates the Q-values by applying the incremental estimation-based update rule

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha (r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a)),$$

(1.6)

where $r_{t+1}$ is the reward received by the agent from the environment when it moves to state $s_{t+1}$ by taking action $a$ from state $s_t$, $\alpha$ is the learning rate ($\alpha \in [0, 1]$), $r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$ is the target value (or label) and the $Q(s_t, a)$ on the right hand side of the equation is the current prediction of Q-value of state $s_t$ and action $a$ pair. The difference between the target value and the current prediction is called the temporal difference error, which is also known as the prediction error or loss. So, after each action, the update rule essentially changes the current Q-value in the direction of the prediction error so that the error is reduced.

To be able to efficiently and accurately estimate the Q-values of the state space and maximize the cumulative expected reward at the same time, a RL agent must carefully balance exploration of environment with exploitation of knowledge already known. Exploration allows the agent to try and experience new (not previously visited) state-action pairs and estimate their Q-values, which helps the agent to find the optimal policy that maximizes the cumulative expected reward. However, if exploration is carried out without restraint in real life, the agent may end up accumulating low levels of rewards as a consequence of trying too many low-reward state-action pairs. On the other hand, if no exploration is carried out, the agent’s estimation of Q-values may not improve and it may keep following a suboptimal policy that will not maximize the cumulative expected reward.

At any given state, the action taken by the agent depends on its exploration policy (or strategy). Some of the commonly used exploration policies are the greedy, $\varepsilon$-greedy, linear annealed $\varepsilon$-greedy, softmax, interval exploration, Boltzmann, max Boltzmann and Boltzmann Gumbel policies. A greedy
policy is a zero-exploration policy which specifies the agent to greedily choose the action that has the highest Q-value at any given state based on the current estimates. An $\varepsilon$-greedy policy specifies the agent to choose a random action with probability $\varepsilon$ and the exploitative action with probability $1 - \varepsilon$. In a linear annealed $\varepsilon$-greedy policy, the value of $\varepsilon$ is linearly decreased with time (experience) so that the agent performs high exploration at the beginning and high exploitation at the end.

### 1.4.3 Deep Reinforcement Learning: Deep Q-Learning

Many real-world problems have a large and/or continuous state space, where it is impossible to record Q-values for every state and action pair. Furthermore, the agent would not be able to visit (experience) all states and try out all actions to obtain the observed rewards $r_{t+1}$ needed to estimate the Q-values using Q-learning. So, Q-values of state-action pairs that have not been encountered yet needs to be generalized from limited experience. One way of doing this is by using perceptrons to approximate the Q-values. A perceptron is essentially an artificial neuron that consists of three components: input nodes, weights and a output node. The input to the perceptron are a set of features or basis functions based on the state variables and the output of the perceptron is the Q-value corresponding to a particular action at the input state. So, one perceptron is needed for each action in the action space. Combining the idea of approximating Q-values using perceptrons and training the agent with Q-learning resulted in the approximation method known as perceptron Q-learning [Kochenderfer (2015)]. In this approach, the weights of the perceptron are updated using the following rule:

$$
\theta \leftarrow \theta + \alpha (r_{t+1} + \gamma \max_{a'} \theta^T \tilde{B}(s_{t+1}, a') - \theta^T \tilde{B}(s_t, a)) \tilde{B}(s_t, a),
$$

where $\theta$ is the weight vector and $\tilde{B}(s_t, a)$ is the set of basis functions which are the inputs to the perceptron.

An inherent drawback of a perceptron is that it can model only linear functions. However, a set of perceptrons can be combined to form a (artificial) neural network which can approximate highly nonlinear functions. Nonlinearity is introduced in the function approximation of neural networks
using activation functions. Sigmoid, hyperbolic tangent (Tanh) and rectified linear unit (ReLU) are a few commonly used activation functions. According to the universal function approximation theorem, a feedforward neural network with one hidden layer, given sufficient neurons and under mild assumptions on the activation function, can approximate any real continuous function. [Cybenko (1989)] was one of the pioneers in proving this theorem for sigmoid activation functions.

A neural network possesses an input and an output layer with some number of hidden layers between them. During training, the neural network learns the appropriate weights for accurately mapping the input features to the predicted output. The backpropagation algorithm is widely used for training (fitting) feedforward neural networks with the objective of minimizing the loss function, which is given by the temporal difference error in the context of Q-learning. In backpropagation, the key idea is to propagate the error backwards through the neural network one layer at a time. The weights associated with the edges connected between the hidden nodes (in the last hidden layer) and the output node(s) are optimized first, and then the weights associated with the edges connected between the nodes in second last hidden layer and nodes in the last hidden layer are optimized, and so on until the layer of input nodes is reached. This means that the gradient of the loss function is not computed with respect to each individual weight of the neural network all at once, which is the inefficient naive approach, but rather computed one layer at a time. While using the chain rule to compute the gradient, this process helps to avoid redundant calculations of intermediate terms in the chain rule as partial computations of the gradient from one layer are reused in that for the previous layer so that the overall gradient computation is efficient.

In deep Q-learning (DQL), a deep neural network is used to approximate the Q-values [Mnih et al. (2013); Mnih et al. (2015)]. Equivalent to a multilayer perceptron, the deep Q-learning neural network (DQN) has several hidden layers, resulting in a large number of weights as its parameters. Q-learning with backpropagation and mini-batch gradient descent is used to update the weights of the neural network such that the temporal difference error loss function is minimized. The weights are updated after each time step (or each action taken by the agent). The mini-batch comprises of a batch of $M$ experience samples, where each experience sample $m = (s^m, a^m, r^m, s'^m)$. Each experience
sample \( m \) can be interpreted as follows: the agent was initially at state \( s^m \), from where it took action \( a^m \) and consequently moved to state \( s'^m \) and received a reward \( r^m \). These samples are generated as the agent interacts with the environment. The experience samples in the mini-batch are drawn randomly from an experience replay memory storing the agent’s past experience samples. At a time step \( t \), the loss function is

\[
L_t(\theta_t) = \frac{1}{2M} \sum_{m=1}^{M} (y^m(r^m, s'^m; \theta_t) - Q(s^m, a^m; \theta_t))^2, \tag{1.8}
\]

where,

\[
y^m(r^m, s'^m; \theta_t) = r^m + \gamma \max_{a'} Q(s'^m, a'); \tag{1.9}
\]

is the target associated with experience sample \( m \) and \( \theta_t \) is the neural network weight vector at time step \( t \). To update the weights at each time step \( t \), gradient descent is applied:

\[
\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_{\theta} L_t(\theta_t). \tag{1.10}
\]

After convergence is reached during training, the optimal policy can be found using Eq. 1.5. As DQL involves the use of a deep neural network, it is considered to be a type of DRL algorithm. In this doctoral research, DQL is primarily used for training the DRL agent of the DeepARM system. A few variants of the DQL algorithm, namely, deep double Q-learning [Van Hasselt et al. (2016)] and DQL with dueling DQN [Wang et al. (2016)], have also been tested.

1.5 Applying Deep Reinforcement Learning to Airline Revenue Management

The ARM problem is a sequential decision-making problem which can be formulated as a MDP. The states may be represented by variables that inform the agent of the current number of bookings and cancellations made in the different fare classes and the time remaining till flight departure. At each decision-making state, the agent may be allowed to open or close fare classes to control seat inventory and vary the prices of the fare products. As commercial airlines aim to maximize flight revenues, the reward the agent receives at each state from taking a certain action may be specified
to be revenue from fares paid by passengers for their bookings minus any fare reimbursements due to cancellations in the time period between two successive states in the state trajectory. In such a formulation, the state would evolve on the basis of the number of bookings and cancellations of the different fare classes and time progression.

The implementation of a DRL-based ARM system in real life would first involve training the DRL agent in an air travel market simulator. The simulator would have to be built based on the airline’s existing demand model, passenger arrival, choice and cancellation models, and competition model. In practice, airlines estimate these models using historical market and booking data. The historical data used by airlines in real-life typically include one or more of the following: 1) proprietary in-house booking data that is generated over time from the airline’s ARM system’s interactions with passengers; 2) commercial data spanning many decades purchased from airline data providers such as International Air Transport Association (IATA), Airline Reporting Company (ARC), Global Distribution Systems (GDS), etc.; and 3) publicly available aggregate data (e.g., O&D Survey DB1A/B and T-100 Reports) spanning two or more decades from government agencies such as US DOT [Holloway (2008)]. The size, accuracy and completeness of the data sets used by the airlines will determine the accuracy of the model estimates, and, in turn, the fidelity of the simulator. As the simulator can be used to potentially generate an unlimited number of data points, the amount of data needed to initially train the DRL agent offline in the simulator is not an issue. Once the DRL agent has been trained to learn the optimal ARM policy of the simulated market in the simulator, it can be deployed on the real world market to directly interact with passengers. No external model or data would be required any further. At this stage, it is essential to ensure that the DRL agent’s hyperparameters is tuned or optimized so that it learns efficiently from the samples of experience (revenue signal, number of bookings, and other observations) drawn from its interactions with the market. The agent would need to balance exploitation (of its current policy) and exploration (of new policies) to keep learning from its interactions and continue updating its ARM policy (neural network weights) accordingly. This continuous learning would allow the agent to learn the true market dynamics and adapt to changes in the market.
1.6 Dissertation Structure

The remainder of this dissertation is organized as follows. Chapter 2 provides a literature review and background of the traditional ARM methods. Chapter 3 describes the application of DRL to the problem of seat inventory control and presents the results of the numerical experiments conducted to evaluate the robustness and revenue performance of the proposed approach. Similarly, Chapter 4 covers the application of DRL to the problem of dynamic pricing and presents the results of the numerical experiments conducted to evaluate the strength of the proposed approach. Finally, the key research findings and promising extensions of this work are discussed in Chapter 5.
CHAPTER 2. TRADITIONAL AIRLINE REVENUE MANAGEMENT METHODS: LITERATURE REVIEW AND BACKGROUND

Since the time of its first application in the airline industry around four decades ago, the field of ARM has been an active research area, leading to continuous improvements in ARM methods and a steady expansion of the field’s body of literature. This interest has been primarily fueled by the success of ARM systems in improving airline revenues by between 2 percent and 8 percent [Li and Peng (2007)]. The use of different ARM approaches, formulations and assumptions for solving the ARM decision-making problem have been reported in the literature. All of these approaches employ one or more of the following three solution methods: mathematical programming, dynamic programming and heuristic rule-based decision models [Talluri and Van Ryzin (2006)]. Without exception, the objective of all these approaches is to maximize the total (expected) flight revenues given a fixed booking period and subject to the constraint of limited seat capacity in each flight. Some of the assumptions commonly made in these approaches are: 1) the arrival distribution of passengers follows either a regular (homogeneous) Poisson process with a constant mean arrival rate or a non-homogeneous Poisson process (NHPP) with a time-varying mean arrival rate [Weatherford et al. (1993), Gallego and Van Ryzin (1994)]; 2) the passenger WTP probability distribution is known; 3) demands for each fare class are separate and independent; 4) lower fare class passengers arrive earlier than higher fare class ones; and 5) there is no competition in the O-D markets [Talluri and Van Ryzin (2006)]. A literature review and background of these traditional approaches and assumptions is presented in this chapter.
2.1 Mathematical Programming

The problem of ARM has been formulated as mathematical programs in several different ways and solved using various optimization techniques by many researchers to date. Over time, the models developed have become more theoretically complex, capturing many real-world problem aspects such as passenger choice behavior and competition. One of the earliest well-known mathematical programming model for ARM was proposed by Glover et al. (1982). He formulated a deterministic integer programming model for determining the revenue-maximizing number of seats to allocate to each fare class offered by the airline in the O-D markets of its route network, as given below.

$$\text{max } \sum_{odf} f_{odf} x_{odf}$$

s.t. \( x_{odf} \leq E[D_{odf}] \quad \forall odf, \)  
$$\sum_{odf \in S_l} x_{odf} \leq \kappa_l \quad l = 1, \ldots, L,$$
$$x_{odf} \geq 0 \quad \forall odf.$$  

(2.1)

In this model, \( odf \) stands for origin-destination-fare class and it indexes all the available fare classes in the various O-D markets of the airline’s route network; and \( f_{odf}, E[D_{odf}] \) and \( x_{odf} \) denote the fare, expected demand and number of allocated seats associated with fare class \( odf \) respectively, \( \kappa_l \) the flight capacity of leg \( l \), \( L \) the total number of flight legs and \( S_l \) the set of fare classes associated with leg \( l \). The objective is to maximize the total network revenue. The decision variables are the \( x_{odf} \)'s, which are constrained to be less than the corresponding fare class expected demand \( E[D_{odf}] \). For any given flight leg \( l \), the number of seats allocated to its associated fare classes are also constrained to be less than the leg’s flight capacity \( \kappa_l \). This model can be extended to include flights operating in the same O-D market at different times of the day.

There are four major drawbacks of this model. Firstly, the model is computationally challenging to solve the model due to the integrality constraints. Its linear programming relaxation is computationally easier to solve, but this approach provides approximate non-integer solutions even when the demand forecasts are integers. Secondly, the model assumes the fare class demands to be
(a) A partitioned three fare class inventory structure 
(b) A nested three fare class inventory structure

Figure 2.2: Types of seat inventory structure

deterministic which is not the case in practice. Thirdly, the static nature of the model does not capture the dynamic nature of the demand; in other words, it cannot respond to actual demand realizations during the booking period. Lastly, the resulting booking limits generated by the model partitions the flight capacity of the airline network, as shown in Fig. 2.1a, such that the sum of the booking limits is equal to the flight capacity. Such a partitioned booking class allocation results in loss of potential revenue as it does not allow high fare class passengers to book seats allocated to lower fare classes once the booking limit of the high fare class is reached. To prevent this from happening, airlines nowadays use nested inventory structures, such as the one shown in Fig. 2.1b [Bertsimas and De Boer (2005)]. In a nested inventory structure, the goal is not to find the number of seats to allocate to all the fare classes, but rather to find the number of seats of higher fare classes that need to be “protected” from the lower fare classes. Note that, unlike the case for partitioned seat inventory structures, the booking limits in a nested structure adds up to be more than the flight capacity.

Wollmer (1986) modified Glover’s model to consider stochastic demand in the problem by using a set of binary decision variables for each fare class. As the number of decision variables increases many fold in his formulation, the model was less attractive for use in practice. To overcome this challenge of dimensionality, de Boer et al. (2002) proposes a stochastic programming model and
Talluri and Van Ryzin (1999) a randomized linear programming model. In addition to demand uncertainty, nested fare class inventory structure was also considered in the two-step approach developed by Curry (1990). In the first step of Curry’s approach, a linear programming model is solved to determine the seat allocations for each itinerary in the network. These itinerary seat allocations are then used to compute nested booking limits for each fare class.

A classic model for the ARM dynamic pricing problem in the literature is the one developed by Gallego and Van Ryzin (1994). In their model, the passenger arrival process is considered to follow a Poisson process and demand varies with price. The objective is to determine the optimal dynamic pricing policy for a single fare product in a given flight - the price to charge passengers at each time step within the booking period such that the expected flight revenue is maximized. The authors derived optimal solutions for a class of exponential demand functions by applying the constrained optimization technique to the model. A deterministic relaxation of their model can handle more general demand functions. The authors subsequently extended their model to address the dynamic pricing problem of network ARM with multiple fare products [Gallego and Van Ryzin (1997)].

An adaptation of Gallego and van Ryzin’s model for the case of demand following a NHPP, as reported in [Selcuk and Avsar (2019)], is given below. This model can be made dynamic by solving it repeatedly at the start of each time step along the booking period to take into account the current demand realization. In this model, \( i \) and \( j \) indexes the time steps and price points respectively; \( p_j \) denotes the price point \( j \), \( y_{ij} \) the number of seats to be sold in time step \( i \) at price point \( j \), \( \kappa \) the remaining flight capacity, \( \mu_{ij} \) the expected demand in time step \( i \) at price point \( j \); and \( x_{ij} \) indicates the availability of price point \( j \) in time step \( i \). The decision variables are \( x_{ij} \) and \( y_{ij} \). The first constraint is the seat capacity constraint. The second constraint ensures that only one price point is selected for the fare product at any given time step. The third constraint ensures that \( y_{ij} \) is zero when \( x_{ij} \) is zero, and it is less than or equal to \( \mu_{ij} \) when \( x_{ij} \) is equal to one.
\[
\begin{align*}
\text{max} & \quad \sum_{i=1}^{\tau} \sum_{j=1}^{m} p_j y_{ij} \\
\text{s.t.} & \quad \sum_{i=1}^{\tau} \sum_{j=1}^{m} y_{ij} \leq \kappa, \\
& \quad \sum_{j=1}^{m} x_{ij} = 1 \quad \forall i = [1, \tau], \\
& \quad 0 \leq y_{ij} \leq \mu_{ij} x_{ij} \quad \forall i = [1, \tau], j = [1, m], \\
& \quad x_{ij} \in 0, 1 \quad \forall i = [1, \tau], j = [1, m]. 
\end{align*}
\] (2.2)

Many dynamic pricing models found in the literature are essentially extensions of Gallego and van Ryzin’s classic model that are capable of handling more complex demand functions and passenger choice behavior [Elmaghraby and Keskinocak (2003); Zhao and Zheng (2000); and Bitran and Mondschein (1997)]. At the same time, there are also models based on entirely different formulations and approaches. Wollmer (1992) developed a stochastic model to determine when to stop accepting booking requests for a seat at a given fare level to save the seat for a potential request at a higher fare level. Using the theory of optimal stopping time and intensity control, Feng and Xiao (2000), and Feng and Gallego (2000) address the dynamic pricing problem by determining the optimal continuous time price switching policy based on the remaining time and seat supply given a set of predetermined finite set of prices at certain time thresholds. Li (2001) finds the optimal pricing policy for the case of three and fewer number of price points by solving a series of linear programming problems.

### 2.2 Dynamic Programming

Many researchers have used dynamic programming (DP) to determine the optimal ARM policy and investigate its structural properties. In this approach, the ARM problem is first formulated as a MDP or a semi-MDP (SMDP). A SMDP is a MDP where the state transitions may take varying lengths of time to occur. In all MDP formulations of the ARM problem found in the literature, the transition function is a function of the passenger booking arrival probability, choice probability, and
cancellation and no-show probability; and the reward function is a function of the fares or revenue values of the fare classes, the cancellation fees, bumping cost, and the number of fare class arrivals, cancellations and passengers bumped. So, the differences in these formulations mostly arise from the choice of the passenger arrival, choice and cancellation models and other problem parameters adopted by the researchers. With the transition and reward functions defined and other problem parameters specified, the value function of the optimal policy is computed using the Bellman equation. To solve the MDP, researchers have used numerous exact DP solution methods, such as value iteration, policy iteration and backward induction.

Van Slyke and Young (2000), in his approach, first relates the problem of ARM to the knapsack problem and then finds a solution to the problem using DP. In this approach, the knapsack is analogous to the aircraft, the objects to the passengers, the weights of the objects to the number of seats requested by the passengers, and the values of objects to the WTPs of passengers. The objective of the problem was to fill a knapsack of fixed capacity with objects of known weights and value such that total value is maximized. He formulated the problem as a discrete state, finite horizon, continuous time DP problem and allowed the airline to accept and reject booking requests. Sawaki (2003) used a similar approach, but he formulated the problem as a SMDP. Chatwin (2000) applied a DP approach to the dynamic pricing problem for determining the optimal continuous time price switching policy.

A stochastic DP-based dynamic pricing model was proposed by Otero and Akhavan-Tabatabaei (2015). They solved the Bellman equations of their model using backward induction. Brumelle and McGill (1993), Robinson (1995), Lee and Hersh (1993), and Lautenbacher and Stidham Jr (1999) have also proposed different stochastic DP-based dynamic pricing models for ARM. Selcuk and Avsar (2019) also formulated the ARM dynamic pricing problem as an exact DP problem and then solved for the optimal pricing policy using backward recursion and gleaned interesting insights by investigating the structural properties of the optimal pricing policy. Lee and Hersh (1993) explored the seat inventory control pertaining to airlines. The problem of optimally deciding on booking requests for a booking class at a specific time was investigated. Discrete-time DP was implemented
to arrive at an optimal policy. As mentioned before, such an approach becomes impractical when the number of states increases or if time is treated as a continuous state variable in the problem. Howard (1960) addressed the overbooking problem assuming the airline did not divide the cabin into different fare classes. The problem was modelled as a MDP and the optimal policy was found using the value iteration algorithm. However, the computational limitations of the value iteration algorithm made this technique unfeasible to implement on large-scale problems. Subramanian et al. (1999) also addressed the seat allocation problem for several fare classes while taking into consideration the possibility of overbooking, cancellations and absentees on the day of the flight. The problem was modeled as a discrete time MDP and an exact solution was found using DP through backward induction. The algorithm was implemented on a real-life airline dataset confirming its computational feasibility. However, their model was based on the assumption that probability of cancellations was not dependent on fare-classes. Gosavi et al. (2002) et al. formulated a similar problem with two major differences. They did not assume that cancellation was independent of fare classes. Additionally, the problem was modeled as a SMDP instead of a MDP. They developed a novel algorithm $\lambda$-SMART to solve the SMDP. The algorithm was compared against the EMSR method and it was found to outperform EMSR. To address the problem of scalability in exact DP models, an approximate DP approach was proposed by Bertsimas and Popescu (2003), where the value function of the DP value function is approximated by Glover’s model.

### 2.3 Heuristic Rule-Based Decision Model: Expected Marginal Seat Revenue

Besides optimization models and dynamic programming-based models, ARM researchers have also developed heuristic rule-based decision models or control frameworks. The most commonly used such decision model in ARM for determining the fare class booking limits on a single flight leg is a heuristic method called the Expected Marginal Seat Revenue (EMSR) method. The objective of the EMSR method is to maximize the total expected revenue of each individual flight leg in the network. As its name suggests, this method’s computations is based on the expected marginal revenue obtained by protecting an additional seat of a certain fare class from all the lower fare classes relative
to it. The first EMSR-based decision models were developed by Littlewood (1972) and Richter (1982) which could handle only two fare classes. Belobaba’s PhD research [Belobaba (1987)] led to the generalization of this approach for multiple fare classes, where he first developed the model known as EMSRa. Later, Belobaba (1989) refined his original model to construct the EMSRb model, which was subsequently extended to include sell-ups [Belobaba and Weatherford (1996)]. Bodily and Weatherford (1995) also derived decision rules to determine when to curtail discount sales based on probability of spoilage and EMSR. To conduct network ARM, the EMSRb method can be used in combination with DAVN to determine booking limits for all virtual fare classes offered in the airline network. Despite its simplicity, the EMSRb method is widely used in the airline industry as a leg-based ARM method because of its robustness and revenue performance [Belobaba (2016)].

Bodily and Weatherford (1995) derives decision rules to determine when to curtail discount sales based on probability of spoilage and EMSR. Brumelle and McGill (1993) et al. tackled the seat allocation problem for several fare classes using the EMSR technique. The problem was formulated on the assumption that ticket requests for high fare classes are placed after the requests for lower fare classes have been made. The rest of the discussion in this section focuses on the EMSRb method, which was used as a performance benchmark in this doctoral research.

The EMSRb model was designed for use with nested fare class structures. The model assumes that fare class demands are independent and fare class passengers arrive sequentially in increasing order of fare classes (as in the lowest fare class passengers arrive first, then the next higher fare class passengers arrive, and so forth).

To determine the optimal fare class booking limits, the EMSRb model requires three inputs: the authorized capacity, the revenue value or fare of each fare class and the probability distribution of demand forecasts of the fare classes. The authorized capacity specifies the maximum number of bookings that may be accepted for the flight and it is typically greater than the physical flight capacity to allow flight overbooking in anticipation of passenger no-shows at flight departure. If passenger no-shows are not considered, then the authorized capacity is simply equal to the flight capacity. The probability distribution of demand forecasts is typically modeled with a normal distribution that best
fits the available historical demand data with the mean set equal to the average demand. Using these inputs, the EMSR from protecting \( s_i \), the \( s \)-th seat of fare class \( i \), can be computed using:

\[
EMSR(s_i) = \overline{f_i} \cdot P(d_i > s_i),
\]

where \( \overline{f_i} \) is the demand-weighted average of revenue values of fare class \( i \) and all higher fare classes, \( d_i \) is the demand of fare class \( i \), and \( P(d_i > s_i) \) is the probability that \( d_i > s_i \). The fare class index \( i \) ranges from 1 to \( I \), where fare class 1 is considered to be the highest fare class and fare class \( I \) the lowest one. If we let \( D_i \) denote the mean demand of fare class \( i \) and \( f_i \) the revenue value of fare class \( i \), then, for any given fare class \( i \),

\[
\frac{\sum_{j=1}^{i} D_j f_j}{\sum_{j=1}^{i} D_j}
\]

(2.4)

To find the optimal number of seats to protect for fare classes 1 through \( i \) from the fare class \( i + 1 \) (\( \rho_i \)), \( s_i \) is increased until

\[
EMSR(s_i = \rho_i) = f_{i+1}
\]

(2.5)

This is a point of indifference at which the EMSR from protecting \( s_i \) is equal to the revenue value of the immediate lower fare class. Note that \( \rho_i \) needs to be calculated for all fare classes except the lowest fare class. Once the \( \rho_i \)'s have been found, the booking limit of a fare class \( i \) is determined by:

\[
BL_i = \tilde{\kappa} - \rho_{i-1},
\]

(2.6)

where \( \tilde{\kappa} \) is the authorized capacity. Note that \( BL_1 = \tilde{\kappa} \) so that the passengers of the highest fare class is never turned away from the booking system in favor of lower fare class passengers as long as there is room for more bookings.
2.4 Summary

A literature review and background of the traditional approaches and common assumptions used for finding the optimal policy for ARM is presented in this chapter. The ARM approaches can be grouped into three categories: mathematical programming, dynamic programming, and heuristic methods. All of these methods require knowledge of market dynamics and passenger behavior in the form of distributions and mathematical models. As a result, their performance depends on the accuracy of these external forecast models and estimated distributions. Moreover, they lack the capability of adapting to changes in market conditions and learning the true dynamics of the problem. Additionally, rule-based heuristics methods, such as EMSRb, do not guarantee optimal or even near-optimal solutions. Lastly, mathematical programming and dynamic programming-based approaches suffer from the curse of dimensionality as the number of decision variables and state space explode with the scale of the problem for these respective approaches. In theory, DRL has the potential to overcome all these limitations for the following reasons: 1) it is a model-free decision framework, so its performance does not depend on the accuracy of external model estimates once it has been trained and starts interacting with the real market; 2) the DRL framework employs a deep neural network to approximate the expected optimal revenues for all possible state-action combinations, allowing it to handle the large state spaces of the problems; and 3) the framework allows exploration, and, hence, enables directly learning the true optimal policy from interactions with the market and adapting its policy to changes in the market based on the reward (revenue) feedback signals from the market and state transitions within the market.
CHAPTER 3. SEAT INVENTORY CONTROL USING DEEP REINFORCEMENT LEARNING

This chapter details the application of DRL to the problem of seat inventory control. First, the problem is formally defined, after which a description of the air travel market simulator used for performing the computational experiments is given. Then, the chapter discusses how the problem was formulated as a MDP, the parameter settings and implementation of the solution method, and the results for the different test cases. Finally, a summary of the research findings is given.

3.1 Problem Statement

The goal of the seat inventory control problem considered in this work is to determine the optimal seat inventory control policy so that the total expected flight revenues are maximized. A seat inventory control policy specifies which fare classes offered in the flight to keep open and which ones to keep closed at the beginning of each time step (decision-making instance) during the booking period. In other words, the objective is to book the optimal number of passengers of each fare class in every flight so that the total expected flight revenues are maximized. A single flight leg with three virtual fare classes with stochastic demands, a booking window of 182 days, and a flight capacity ($\kappa$) of 100 have been considered in the problem. The three virtual fare classes will simply be referred to as the high (H), middle (M) and low (L) fare class in the remainder of this chapter. The set of all fare classes is denoted by $I$, and, hence, $I = \{H, M, L\}$. The subscript $i$ will be used as the index of fare class, and, hence, $i \in I$. Their respective average revenue values, denoted by $f_H, f_M$ and $f_L$, are considered to be $400, 200$ and $100$. The chosen values resemble the typical one-way fares of a short-haul domestic flight of a low cost carrier in the US. Both high and middle fare class passengers
may cancel their bookings free of cost. This implies that the optimal seat inventory control policy must also carry out flight overbooking optimally. The fare class booking arrival and cancellation distribution parameters are described in Sections 3.2.1 and 3.2.2 below. The booking window is split into 182 days or time steps of equal duration. At the beginning of each time step, the airline can decide to keep each of the fare classes open or closed for the duration of the time step. Passengers can book seats of a certain fare class only if it is open. Before a flight departure, the airline can overbook the flight in anticipation of future cancellations. As the problem involves taking a series of actions at different points in time till the date of departure, this problem is a sequential decision-making problem.

Taking the above problem specification into account, the seat inventory control problem is suitable for being modeled as a MDP. In this work, we formulate this problem as a MDP and then find the optimal seat inventory control policy by using a DRL approach. We conduct several numerical experiments to analyze the performance of our approach in different market settings.

### 3.2 Air Travel Market Simulator

It is a standard practice among ARM practitioners and researchers to use air travel market simulators to test the theory and analyze the performance, such as revenue and load factor impacts, of novel ARM techniques. The simulators are designed to model passenger arrival, choice and cancellation behavior and the interactions of passengers with the ARM system(s) in airline markets. One such well-known air travel market simulator in the ARM community is the Passenger Origin-Destination Simulator (PODS), originally conceived at Boeing and currently maintained by the PODS Research LLC [PODS Research, LLC. (2017)]. This simulation environment tool is heavily utilized for ARM research by the members of the MIT/PODS Research Consortium, which as of 2016 included the following major airlines and airline IT solution vendors: Air Canada, Amadeus, American Airlines, ATPCO, the Boeing Company, Delta, Emirates, Etihad, LATAM, Lufthansa/Swiss, PROS, Qatar Airways, SABRE and United.

For training and testing our DRL agent, we have also created an air travel market simulator that
generates flight episodes simulating passenger booking arrivals and cancellations of different fare classes at the different time steps of the booking period. The set of all flight episodes is divided into two sets, a training set and a test set. Our simulator consists of two parts: a passenger arrival model and a booking cancellation model. Following the problem statement, simulated passengers are allowed to book seats at most 182 days prior to the flight departure. Upon arriving, passengers are assumed to either book their seats if their desired fare class is open or leave the booking system if their desired fare class is closed at the current time step. High and middle fare class passengers have the option of canceling their bookings at any time step following that in which they made the booking. Based on the knowledge of passenger booking arrivals and cancellations, the simulator computes the theoretical optimal revenue that may be generated in each flight episode by optimally controlling the seat inventory. The revenue performance of the DRL agent is assessed relative to this theoretical optimal revenue. The passenger arrival and cancellation models and the computation of the theoretical optimal revenue are described in the following sections.

3.2.1 Passenger Arrival Model

The passenger arrival of each fare class has been modeled as a NHPP, a commonly used approach in the ARM literature as NHPPs have been shown to fit well real-world airline passenger arrival data [Beckmann and Bobkoski (1958); Weatherford et al. (1993); Bertsimas and De Boer (2005)]. Unlike a regular Poisson process, in a NHPP, the mean arrival rate changes with time, as does the mean arrival rate of passengers of different fare classes in real life. The time-varying mean arrival rate of the different fare classes are determined using

$$\lambda_i(t) = \int_{t}^{t+1} \lambda_i'(\tau; \hat{\alpha}_i, \hat{\beta}_i) \hat{A}_i d\tau,$$

where $\lambda_i(t)$ is the mean arrival rate of fare class $i$ at time step $t$, $\lambda_i'(\tau; \hat{\alpha}_i, \hat{\beta}_i)$ is the probability density function value of a beta distribution with shape parameters $\hat{\alpha}_i$ and $\hat{\beta}_i$ corresponding to fare class $i$ at time $\tau$, and $\hat{A}_i$ is the mean number of arrivals of fare class $i$ in each flight episode.
The use of the beta distribution allows us to specify different types of passenger arrival trends using its shape parameters. The probability density function of the beta distribution is defined as

$$
\lambda_i'(\tau) = \frac{(\tau \tau_{182})^{\hat{\alpha}_i - 1} (1 - \tau \tau_{182})^{\hat{\beta}_i - 1}}{B(\hat{\alpha}_i, \hat{\beta}_i)} \tag{3.2}
$$

where

$$
B(\hat{\alpha}_i, \hat{\beta}_i) = \frac{\Gamma(\hat{\alpha}_i) \Gamma(\hat{\beta}_i)}{\Gamma(\hat{\alpha}_i + \hat{\beta}_i)} \tag{3.3}
$$

and \(\Gamma\) is the Gamma function. As the length of the booking period is 182 days, \(t \in \{0, ..., 182\}\) and \(\tau \in [0, 182]\). Note that \(t = 182\) occurs at the end of the booking period (or flight departure). During simulation, the number of arrivals of fare class \(i\) at each time step \(t\) \((A_{t,i})\) is determined by sampling from a Poisson distribution with the mean arrival rate \(\lambda_i(t)\) (i.e., \(A_{t,i} \sim \text{Poisson}(\lambda_i(t))\)). The mean number of arrivals of each fare class in different numerical experiments varies with the mean arrival setting (MAS) as shown in Table 3.1. The total number of expected passengers is between 150 and 156 in each MAS. Having this number exceed the flight capacity in each MAS ensures that the naive policy of keeping all fare classes open throughout the booking period is not an optimal one in most episodes. In the traditional NHPP-based passenger arrival model, \(\hat{A}_i\) is replaced by the total number of passenger arrivals, which is typically modeled with a gamma distribution. In our model, we have kept \(\hat{A}_i\) deterministic. The shape parameters \(\hat{\alpha}_i\) and \(\hat{\beta}_i\) of the beta distributions determine the passenger arrival patterns of the fare classes. Table 3.2 lists the shape parameter values used in the simulator.

### Table 3.1: The fare class mean arrivals per episode in different mean arrival settings

<table>
<thead>
<tr>
<th>MAS</th>
<th>(\hat{A}_H)</th>
<th>(\hat{A}_M)</th>
<th>(\hat{A}_L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>40</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>40</td>
<td>80</td>
</tr>
</tbody>
</table>

Figure 3.1 shows how the arrival rate of each fare class varies with time throughout the booking period for MAS equal to 2. As evident from the figure, a NHPP-based passenger arrival model has
a number of desirable properties: 1) most lower fare class passengers have a greater likelihood of arriving earlier than higher fare class passengers; 2) the order of arrivals is uncertain; 3) the number of passenger arrivals of each fare class in each flight episode is uncertain; and 4) the mix of passenger arrivals of the fare classes is uncertain.

### 3.2.2 Cancellation Model

Typically, passengers are more likely to cancel their bookings right after they made one and shortly before departure, resulting in a tub-shaped cancellation rate graph. In our approach, we have assumed that the passenger cancellation rates are only high near the end of the booking period. For the high and middle fare classes, the passenger cancellation rate is considered to be 0.01% at all time steps preceding the last two time steps. In the last two time steps ($t \in \{180, 181\}$), the

---

**Figure 3.1:** Plot of passenger arrivals rates of different fare classes for the second MAS

**Table 3.2: The beta distribution shape parameter values of the fare classes**

<table>
<thead>
<tr>
<th>Fare class ($i$)</th>
<th>$\hat{\alpha}_i$</th>
<th>$\hat{\beta}_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>28.35</td>
<td>1.65</td>
</tr>
<tr>
<td>M</td>
<td>15.4</td>
<td>4.6</td>
</tr>
<tr>
<td>L</td>
<td>11.55</td>
<td>18.45</td>
</tr>
</tbody>
</table>
cancellation rate is determined by the cancellation probability setting (CPS). The different CPSs and their corresponding fare class cancellation rates are listed in Table 3.3. Similar cancellation modeling approaches are commonly used in the industry and literature as they closely resemble real-world passenger booking cancellation behavior [Iliescu et al. (2008), Petraru (2016)].

Table 3.3: The fare class cancellation rates in different cancellation probability settings

<table>
<thead>
<tr>
<th>CPS</th>
<th>H</th>
<th>M</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>10%</td>
<td>5%</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>15%</td>
<td>10%</td>
<td>0%</td>
</tr>
</tbody>
</table>

3.2.3 Optimal Revenue Computation

The optimal revenue of any given flight is the maximum possible revenue that can be generated in that flight. In other words, it is the upper bound of the flight revenue. Given the total number of passenger arrivals and cancellations of each fare class in a flight episode \(e\), the maximum possible revenue that can be generated from booking passengers in that flight can be computed using

\[
r^*_e = \sum_{i \in I} n^*_{i,e} f_i,
\]

where \(r^*_e\) is the optimal revenue in flight episode \(e\), \(n^*_{i,e}\) is the optimal number of passengers to book from fare class \(i\) in flight episode \(e\), and \(f_i\) is the fare (revenue value) of fare class \(i\). In theory, \(r^*_e\) will be generated by accommodating first the committed passengers (passengers who will not cancel after making a booking) from the highest fare class in the flight, and then followed by the committed passengers of lower fare classes in descending order of fare until the capacity is filled or all passengers are accommodated. So, \(n^*_{i,e}\) can be computed using

\[
n^*_{i,e} = \min \left( A^T_{i,e} - C^T_{i,e}, \tilde{\kappa}_{i,e} \right),
\]

where
\[ \hat{\kappa}_{i,e} = \kappa - \sum_{i \in \{H, \ldots, i-1\}} n_{i-1,e}^{*}, \]  

(3.6)

where \( A_{i,e}^{T} \) is the total number of passenger arrivals of fare class \( i \) in episode \( e \), \( C_{i,e}^{T} \) is the total number of passengers of fare class \( i \) who will cancel their booking at some point before flight departure in episode \( e \), and \( \hat{\kappa}_{i,e} \) is the remaining flight capacity after the optimal number of passengers from fare classes higher than fare class \( i \) has been accommodated. The number of committed passengers of fare class \( i \) in flight episode \( e \) is equal to \( A_{i,e}^{T} - C_{i,e}^{T} \). Fare class \( i-1 \) is the immediate higher fare class with respect to fare class \( i \). For the highest fare class, \( \hat{\kappa}_{H,e} = \kappa \) as no passengers of any other fare class have been booked yet. Serving as the upper bound, this optimal flight revenue is used to benchmark the performance of the agent during the training and testing phases. More specifically, the agent’s performance is measured in terms of the average percentage of optimal revenue it is able to generate in the flight episodes at the end of training and during testing. Note that it is not possible for any ARM system to generate this optimal revenue in most flight episodes as the complete and exact knowledge of the total number of passenger arrivals and cancellations of each fare class in a flight episode is not known ahead of time at the beginning of a flight episode.

### 3.3 Markov Decision Process Formulation

The primary components of the MDP formulation of our ARM problem has been defined in this section based on the problem description given in Section 3.1.

#### 3.3.1 States

Each state \( s \) in the state space \( S (s \in S) \) is represented by a vector of integer state variables to capture the booking activity and current time information. A state \( s \) is defined as

\[ s = (b_H, b_M, b_L, t), \]  

(3.7)

which includes three integer booking count variables \( b_H, b_M \) and \( b_L \) that keep track of the number of bookings made of the high, middle and low fare classes respectively, and the time step variable
\( t \) to denote the current time step in the booking period. As an example, a state \( s = (5, 30, 45, 170) \) denotes that the current time step is 170 and the present number of high, middle and low fare class bookings are 5, 30 and 45 respectively. The representation of the states in this way results in a state space of size in the order of approximately higher than \( 10^9 \) if the Poisson arrival distributions are truncated at the points beyond which the arrival probabilities are less than the order of \( 10^{-2} \). Note that the states can also be represented as \( s_t \), as was done in Section 1.4.1.

### 3.3.2 Actions

At the beginning of each time step, the agent can take an action \( a \) to specify which fare classes will stay open and which ones will stay closed during the current time step. The actions constitute the action space \( A (a \in A) \). The set of allowed actions are

\[
a \in \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}.
\]

The interpretation of the different actions is given in Table 3.4. Action \( \hat{a}_1 \) specifies that only the high fare class is open and the other two lower classes are closed. Action \( \hat{a}_2 \) specifies that both the high and middle fare classes are open and the lowest fare class is closed. Finally, action \( \hat{a}_3 \) specifies that all fare classes are open. In all the different actions, the high fare class is always open as it is never an optimal action to turn away high fare class passengers in favor of lower class passengers. Also, there is no action in \( A \) which specifies the middle fare class to be closed and the low fare class to be open as such an action is never an optimal action at any given time step. In total, the agent has to take 182 actions in each flight episode before flight departure.

<table>
<thead>
<tr>
<th>Action</th>
<th>( H )</th>
<th>( M )</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{a}_1 )</td>
<td>open</td>
<td>close</td>
<td>close</td>
</tr>
<tr>
<td>( \hat{a}_2 )</td>
<td>open</td>
<td>open</td>
<td>close</td>
</tr>
<tr>
<td>( \hat{a}_3 )</td>
<td>open</td>
<td>open</td>
<td>open</td>
</tr>
</tbody>
</table>
A passenger of a certain fare class can only make a booking if that fare class is open. If a passenger’s desired fare class is closed, the passenger is assumed to leave the booking session. If it is open, the passenger is assumed to make a booking. Action \( \hat{a}_1 \) is useful for saving the remaining seats for the high fare class passengers in the current time step. Similarly, action \( \hat{a}_2 \) allows the agent to save the remaining seats for the high and middle fare class passengers in the current time step. The third action \( \hat{a}_3 \) is useful for increasing the load factor during the initial lean period of passenger arrivals and in flight episodes with low passenger arrivals. Through these actions, the agent can control the seat inventory during flight booking to maximize revenue based on passenger arrivals.

### 3.3.3 Model Dynamics

At the end of each time step, the time step state variable advances by one and the booking count variables get updated based on the the number of new bookings and number of booking cancellations of each fare class that occurred in the time step. For every new booking, the corresponding booking count variable is incremented, and, for every booking cancellation, the corresponding booking count variable is decremented. So, the net change in the booking count variable of fare class \( i \) in time step \( t \) is given by

\[
\Delta b_{t,i} = \beta_{t,i} - C_{t,i}
\]

where \( \beta_{t,i} \) and \( C_{t,i} \) are the total number of new bookings and booking cancellations of fare class \( i \) in the given time step \( t \) respectively. Let \( \alpha_{t,i} \) denote the number of passenger arrivals of fare class \( i \) in time step \( t \). Clearly,

\[
\beta_{t,i} = \begin{cases} 
\alpha_{t,i}, & \text{if fare class } i \text{ is open at } t \\
0, & \text{otherwise}
\end{cases}
\]

Once the terminal state \( (t = 182) \) is reached, all actions will lead to the ending of the episode.
3.3.4 Reward Function and Discount Factor

The reward function $R(s, s')$ specifies the revenue received by the agent at the end of each time step $t$ when it moves from one state $s$ to another state $s'$. It is defined as:

$$R(s, s') = \begin{cases} 
\sum_{i \in \mathcal{I}} \Delta b_{t,i} f_i, & \text{for } 0 \leq t \leq 181 \\
- \sum_{i \in \{L, M\}} \eta_i B f_i - 675 \eta_H, & \text{if } t = 182,
\end{cases}$$

(3.11)

where $s = (b_H, b_M, b_L, t)$, $s' = (b'_H, b'_M, b'_L, t')$ and $\Delta b_{t,i} = b'_i - b_i$. At all time steps before flight departure ($0 \leq t \leq 181$), the revenue generated from each fare class $i$ is $\Delta b_{t,i} f_i$, the total amount resulting from adding up all the fares collected from the fare class bookings and subtracting all the fare reimbursements due to the fare class booking cancellations. Note that, for a fare class $i$, if the number of cancellations is more than the number of bookings in time step $t$, $\Delta b_{t,i}$ will be negative and so will be the revenue from fare class $i$ in time step $t$. The total revenue generated during these time steps is the sum of the individual fare class revenues. At the terminal state ($t = 182$), if the total number of passenger bookings is greater than the flight capacity, the agent will incur a bumping cost for having to deny boarding to some passengers. This bumping cost is equal to $- \sum_{i \in \{L, M\}} \eta_i B f_i - 675 \eta_H$, where $\eta_i$ is the number of passengers bumped from fare class $i$ and $B$ is the bumping cost factor. The total number of passengers that need to be bumped ($\eta^T$) to ensure that the flight load factor is less than or equal to one can be determined using

$$\eta^T = \max(\sum_{i \in I} b_i - \kappa, 0), \quad t = 182.$$  

(3.12)

If $\eta^T$ is found to be greater than zero, the flight is said to be overbooked or oversold. Because the bumping cost is a negative reward, it acts as a disincentive to help the agent learn to avoid overbooking flights to the extent that some passengers have to be bumped at flight departure. The cost of bumping a passenger is considered to be the denied boarding compensation given to the passenger. Following the US Department of Transportation’s denied boarding compensation policy
[US DOT (2020)], for every passenger bumped, the bumping cost incurred is considered to be the fare paid by the passenger multiplied by a bumping cost factor ($B$) of 2 and no more than $675, assuming that bumped passengers experience an arrival delay of 1 to 2 hours. For any fare class $i$, multiplying the per passenger bumping cost with the number of passengers bumped from fare class $i$ ($\eta_i$) gives the total fare class bumping cost. The total bumping cost is simply the sum of the individual fare class bumping costs. When it comes to deciding which passenger(s) to bump, airlines are legally free to choose their own bumping policy that they perceive to be fair [US DOT (2020)]. In our problem, passengers are assumed to be bumped in ascending order of fare classes. In other words, passenger bumping starts with the lowest fare class and keeps moving onto higher fare classes until the remaining number of booked passengers is equal to the flight capacity. The number of passengers to be bumped from fare classes L, M and H ($\eta_L, \eta_M$ and $\eta_H$) can be computed using the following set of equations:

$$\eta_L = \begin{cases} 
\eta^T, & \text{if } \eta^T \leq b_L \\
b_L, & \text{if } \eta^T > b_L
\end{cases} \quad (3.13)$$

$$\eta_M = \begin{cases} 
\eta^T - \eta_L, & \text{if } \eta^T - \eta_L \leq b_M \\
b_M, & \text{if } \eta^T - \eta_L > b_M
\end{cases} \quad (3.14)$$

$$\eta_H = \eta^T - \eta_L - \eta_M \quad (3.15)$$

Assuming the effect of monetary discounting within the booking period to be insignificant, a discount factor of 0.995 was chosen to facilitate convergence during training.

### 3.4 Solution Method

To learn and evaluate a seat inventory control policy for the ARM MDP formulated in the preceding section, a DRL agent was trained and tested with the air travel market simulator. The DRL learning algorithm was chosen to be deep Q-learning due to its simplicity and success in numerous
applications, as discussed in Chapter 1. The interactions taking place between the agent and the market simulator is shown in Fig. 3.2, where $s$, $a$ and $r$ are the present state, action and reward respectively, and $s'$ and $r'$ are the state and reward at the next time step.

### 3.4.1 Neural Network Architecture

A dense, feedforward neural network with three hidden layers was used to approximate the Q-values. The input layer has five nodes, one node for each variable in the state vector and an additional bias node. Each of the three hidden layers are composed of 256 ReLU-activated hidden neurons. The output layer has three linearly activated nodes, one node for each action. The architecture of the neural network is depicted in Fig. 4.4.

### 3.4.2 Hyperparameter Setting

During training, the deep Q-learning algorithm updated the weights of the neural network based on the agent-market simulator interaction samples. A soft weight update approach was used by setting the target model update parameter of DQL to 0.01. The optimization algorithm used was Adam with a learning rate of 0.001. The training batch size was 32. A linear annealed $\epsilon$-greedy policy was chosen as the exploration policy. The value of $\epsilon$, the probability with which the agent
Figure 3.3: The configuration of the deep neural network used to approximate Q-values

takes a random action, was linearly decreased from 1 to 0.01 over the training phase of 25,000 flight episodes.

Several different settings of hyperparameter values were explored before settling on the set of values mentioned above which was found to generate the highest flight revenues among all the choices considered. Variations of deep Q-learning, namely, double DQN and dueling DQN, and other exploration policies, namely, Boltzmann exploration and $\varepsilon$-greedy, were also experimented with while hand-tuning the hyperparameter values. As expected, the performance of the agent is sensitive to the hyperparameter values. So, one interesting future research direction involves using a hyperparameter optimization scheme, such as Bayesian hyperparameter optimization, to further improve or optimize the agent’s performance.

3.5 Results

A total of 9 different numerical experiments were conducted by varying the MAS and CPS to simulate different market settings so that the robustness of the solution method can be evaluated. In
all experiments, the percentage of optimal revenue and the load factor the agent achieved at the end of each flight episode during training and testing were recorded. While training the agent, the weights of the agent’s neural network model were saved at intervals of 250 flight episodes. So, for 25000 training flight episodes, a total of 100 neural network models were saved. Each of these models were tested on a separate unseen set of 300 test flight episodes. Training was considered to converge when the change in optimal revenue percentage moving average is less than 0.05 for 10 consecutive flight episodes. During testing, the neural network weights are not updated and the models follow a greedy policy. A greedy policy instructs the agent to choose the action that has the highest Q-value at the current state. Out of all saved models, the model that gave the best performance in terms of the average percentage of optimal revenue and the average load factor was selected as the final model.

3.5.1 Training Results

The training plots for the numerical experiment with MAS=1 and CPS=1 are presented in Figs. 3.4 and 3.5. Predictably, at the start of training, the agent is unable to perform well as its neural network is initialized with random weights and it is mostly choosing its actions randomly for the sake of exploration of the state-action space. However, as training progresses, the agent learns from its interactions to achieve higher percentage of optimal revenue and bring the load factor down to close to 100% despite not having any domain knowledge of the problem and market dynamics, such as flight capacity, fare class fares, the distribution of passenger arrivals and the cancellation rates of each fare class, etc., initially. Both the plot lines level out near the end of training after convergence is reached. Training plots of other numerical experiments show similar trends.

3.5.2 Testing Results

The best model of the numerical experiment with MAS=1 and CPS=1 was found to score an average optimal revenue of 96.44% and an average load factor of 99.57%. The test plots of this model are presented in Figs. 3.6, 3.7 and 3.8. The red line in the first two plots represents the moving average and the blue dashed line the overall average. Note that it is desirable to have these
Figure 3.4: Average optimal revenue percentage generated by agent during training for MAS = 1 and CPS = 1

Figure 3.5: Load factor achieved by agent during training for MAS = 1 and CPS = 1
Figure 3.6: Optimal revenue percentage generated by agent during testing for MAS = 1 and CPS = 1

Figure 3.7: Load factor achieved by agent during testing for MAS = 1 and CPS = 1
performance metrics as close to 100% as possible, but it is not possible for any ARM system to achieve an average optimal revenue of 100% when the fare class demands and cancellations are stochastic. An average load factor of less than 100% suggests that the agent has learned for the most part to avoid the undesirable state of having to bump passengers from the flight before departure. The booking plot given in Fig. 3.8 shows the variation in the number of passengers of different fare classes booked by the agent in the test flight episodes. The agent can be observed to book on average 14.41, 39.04 and 46.12 out of the average 16, 40 and 100 passenger arrivals of the high, middle and low fare classes respectively in the flight episodes during testing. Note that the sum of the individual average fare class bookings is equal to the average load factor of 99.57%. These results suggest that the agent has learned to optimally control seat inventory in a way that protects seats for later arriving higher class passengers from earlier arriving lower class passengers and overbook flights before departure based on passenger arrival distributions and fare class cancellation rates respectively such that the expected flight revenue is maximized and the load factor is near 100%. Similar test plots can be observed for the best models found in all other numerical experiments.

3.5.3 DRL vs EMSRb

In addition to the percentage of optimal revenue generated, the performance of the EMSRb agent (i.e., an agent following the EMSRb ARM policy) is also used to benchmark the performance of the DRL agent. The EMSRb method is chosen as a benchmark because it is widely used in the airline industry as a leg-based ARM method due to its simplicity, robustness and strong revenue performance as discussed in Chapter 2. In each numerical experiment, both the agents have been tested on the same set of test flight episodes. The test performance of the DRL agent and EMSRb agent in each numerical experiment is reported in Tables 3.5 and 3.6 respectively. The improvement in performance of the DRL agent relative to the EMSRb agent is listed in Table 3.7. The performance metrics include the average percentage of optimal revenue generated (\( \mu_{RP} \)), standard deviation of the percentage of optimal revenue generated (\( \sigma_{RP} \)), average load factor (\( \mu_{LF} \)), standard deviation of the load factor (\( \sigma_{LF} \)), average revenue generated (\( \mu_R \)), standard deviation of the revenue generated...
(σₚ) and the 95% confidence interval for the revenue generated (95% CIₚ).

Overall, the DRL agent can be observed to outperform the EMSRb agent in carrying out the task of seat inventory control and overbooking. In all experiments, the DRL agent achieves an \( \mu_{RP^*} \) value of between 96% and 97%, whereas the EMSRb agent achieves an \( \mu_{RP^*} \) value of between 94.5% and 95.6%. On average, the DRL agent achieves a 1.58% improvement in \( \mu_{RP^*} \) with a \( \sigma_{RP^*} \) that is less by 0.77% relative to that achieved by the EMSRb agent. In terms of revenue, the DRL agent generates a revenue increment of $317.44 in each flight episode on average compared to the EMSRb agent. This means the DRL agent achieves a 1.73% revenue increment per flight on average relative to the EMSRb method. Because airlines typically operate flights in a particular route multiple times per day at different times throughout the year, such a revenue increment translates to a significantly large number at the end of each year when it is summed over all the flights of that year. To illustrate this, if we suppose that the airline is operating two flights daily in the flight leg considered in our problem, the average revenue gain at the end of a year would be $231731.2.

Figure 3.8: Fare class booking plot of agent during testing for MAS = 1 and CPS = 1
Table 3.5: Performance of the DRL agent in the different numerical experiments

<table>
<thead>
<tr>
<th>CPS</th>
<th>MAS</th>
<th>$\mu_{RP}$ (%)</th>
<th>$\sigma_{RP}$ (%)</th>
<th>$\mu_{LF}$ (%)</th>
<th>$\sigma_{LF}$ (%)</th>
<th>$\mu_R$ ($)</th>
<th>$\sigma_R$ ($)</th>
<th>95% CI $R$ ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>96.44</td>
<td>2.68</td>
<td>99.57</td>
<td>8.01</td>
<td>17584.33</td>
<td>1445.43</td>
<td>±164.50</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>96.57</td>
<td>2.51</td>
<td>98.91</td>
<td>8.32</td>
<td>19961.33</td>
<td>1763.64</td>
<td>±200.72</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>96.96</td>
<td>2.24</td>
<td>102.52</td>
<td>7.96</td>
<td>20900</td>
<td>1451.57</td>
<td>±165.20</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>96.35</td>
<td>2.5</td>
<td>104.73</td>
<td>6.75</td>
<td>16928.33</td>
<td>947.79</td>
<td>±107.87</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>96.97</td>
<td>2.12</td>
<td>102.36</td>
<td>7.09</td>
<td>19191</td>
<td>1361.62</td>
<td>±154.96</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>96.65</td>
<td>2.61</td>
<td>98.38</td>
<td>8.3</td>
<td>20007.33</td>
<td>1620.91</td>
<td>±184.47</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>96.73</td>
<td>2.47</td>
<td>100.87</td>
<td>6.96</td>
<td>16102.33</td>
<td>1131.63</td>
<td>±126.74</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>96.76</td>
<td>2.59</td>
<td>97.6</td>
<td>6.94</td>
<td>18134.33</td>
<td>1587.09</td>
<td>±180.62</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>96.71</td>
<td>2.42</td>
<td>104.01</td>
<td>7.22</td>
<td>18865</td>
<td>1152.94</td>
<td>±131.21</td>
</tr>
</tbody>
</table>

Table 3.6: Performance of the EMSRb agent in the different numerical experiments

<table>
<thead>
<tr>
<th>CPS</th>
<th>MAS</th>
<th>$\mu_{RP}$ (%)</th>
<th>$\sigma_{RP}$ (%)</th>
<th>$\mu_{LF}$ (%)</th>
<th>$\sigma_{LF}$ (%)</th>
<th>$\mu_R$ ($)</th>
<th>$\sigma_R$ ($)</th>
<th>95% CI $R$ ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>94.63</td>
<td>3.64</td>
<td>96.42</td>
<td>4.28</td>
<td>17228.67</td>
<td>1165.18</td>
<td>±132.61</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>94.9</td>
<td>3.21</td>
<td>96.09</td>
<td>4.82</td>
<td>19590.67</td>
<td>1491.68</td>
<td>±169.77</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>95.53</td>
<td>3.12</td>
<td>96.29</td>
<td>4.72</td>
<td>20585</td>
<td>1436.11</td>
<td>±163.44</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>95.1</td>
<td>3.23</td>
<td>97.12</td>
<td>3.94</td>
<td>16711.33</td>
<td>1039.23</td>
<td>±118.27</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>95.18</td>
<td>3.16</td>
<td>97.07</td>
<td>4.54</td>
<td>18832.33</td>
<td>1338.98</td>
<td>±152.39</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>95.06</td>
<td>3.36</td>
<td>96.7</td>
<td>4.86</td>
<td>19653.33</td>
<td>1354.56</td>
<td>±154.16</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>94.98</td>
<td>3.23</td>
<td>97</td>
<td>4.3</td>
<td>15804</td>
<td>1064.35</td>
<td>±121.13</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>95.18</td>
<td>3.06</td>
<td>97.23</td>
<td>4.61</td>
<td>17811.33</td>
<td>1259.84</td>
<td>±143.38</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>95.34</td>
<td>3.03</td>
<td>97.33</td>
<td>4.62</td>
<td>18600.33</td>
<td>1205.75</td>
<td>±137.22</td>
</tr>
</tbody>
</table>
Table 3.7: Performance of the DRL agent relative to the EMSRb agent in the different numerical experiments

<table>
<thead>
<tr>
<th>CPS</th>
<th>MAS</th>
<th>$\mu_{RP}^*$ (%)</th>
<th>$\sigma_{RP}^*$ (%)</th>
<th>$\mu_{LF}$ (%)</th>
<th>$\sigma_{LF}$ (%)</th>
<th>$\mu_R$ ($)</th>
<th>$\sigma_R$ ($)</th>
<th>95% CI_R ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.81</td>
<td>-0.96</td>
<td>3.15</td>
<td>3.73</td>
<td>355.66</td>
<td>280.25</td>
<td>31.89</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1.67</td>
<td>-0.7</td>
<td>2.82</td>
<td>3.5</td>
<td>370.66</td>
<td>271.96</td>
<td>30.95</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1.43</td>
<td>-0.88</td>
<td>6.23</td>
<td>3.24</td>
<td>315</td>
<td>15.46</td>
<td>1.76</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.25</td>
<td>-0.73</td>
<td>7.61</td>
<td>2.81</td>
<td>217</td>
<td>-91.44</td>
<td>-10.4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.79</td>
<td>-1.04</td>
<td>5.29</td>
<td>2.55</td>
<td>358.67</td>
<td>22.64</td>
<td>2.57</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1.59</td>
<td>-0.75</td>
<td>1.68</td>
<td>3.44</td>
<td>354</td>
<td>266.35</td>
<td>30.31</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1.75</td>
<td>-0.76</td>
<td>3.87</td>
<td>2.66</td>
<td>298.33</td>
<td>49.28</td>
<td>5.61</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1.58</td>
<td>-0.47</td>
<td>0.37</td>
<td>2.33</td>
<td>323</td>
<td>327.25</td>
<td>37.24</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1.37</td>
<td>-0.61</td>
<td>6.68</td>
<td>2.6</td>
<td>264.67</td>
<td>-52.81</td>
<td>-6.01</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td>1.58</td>
<td>-0.77</td>
<td>4.19</td>
<td>2.98</td>
<td>317.44</td>
<td>120.99</td>
<td>13.77</td>
</tr>
</tbody>
</table>

The $\mu_{LF}$ and $\sigma_{LF}$ of the EMSRb agent is lower than that of the DRL agent by 4.19% and 2.98% on average, which suggests that the EMSRb agent is more conservative in booking low and middle fare class passengers and overbooking passengers in the flights. In other words, the EMSRb agent tends to enforce low booking limits for the low and middle fare class passengers to protect seats for the later arriving high fare class passengers, resulting in a low $\mu_{LF}$. In contrast, the DRL agent scores a slightly higher than 100% $\mu_{LF}$ in some of the experiments, especially the ones in which the mean number of high fare class passenger arrivals is higher (MAS $\geq$ 2) and the cancellation rates are higher (CPS $\geq$ 2). Although, a $\mu_{LF}$ greater than 100% means that the DRL agent is bumping a few passengers at the end of each flight episode on average, the agent is still acting optimally. This can be attributed to the low cost of bumping low fare class passengers specified in the problem definition. For every low fare class passenger bumped, the DRL agent incurs a loss of $100. Note that the problem setup does not allow the seat inventory to be controlled before the arrival of each individual passenger. Instead, the seat inventory control decisions are made at the beginning of each time step, during which a group of passengers of each fare class typically arrive. At each time step, as the DRL agent can either book all the arriving low fare class passengers or turn all of them away, in some cases, keeping the fare class open to book all of them and then bumping a few at the end of the flight
episode generates more revenue than doing the opposite (keeping the fare closed) on average. As a result, the average load factor generated by the DRL agent is slightly greater than 100%.

3.5.4 DRL vs Dynamic Programming: Learning the Optimal Seat Inventory Control Policy

Many researchers in the recent past have derived the optimal policy of the ARM MDP for the small-scale ARM problem using exact DP methods and investigated its structural properties as discussed in Chapter 2. To determine whether a DRL agent is able to learn the optimal policy of the ARM MDP in general, the percentage of optimal revenue generated by following the DRL agent’s policy can be compared to the percentage of optimal revenue obtained by following the optimal policy of the ARM MDP to check how closely they match. The difference in the revenues is a measure of the optimality gap of the DRL agent’s policy. So, the smaller the difference, the closer the policy of the DRL agent is to the optimal policy. The exact ARM optimal policy can be found by using exact DP methods such as value iteration, policy iteration and backward induction. Although, the successful application of DP methods leads to the optimal policy, they are not attractive for use in practice as they require the complete knowledge of the transition function, which is never known exactly in real life, and they cannot handle large discrete or continuous state and/or action spaces due to computational intractability. To carry out this test and determine the difference in performance of the policies, the current ARM MDP problem with 3 fare classes and a flight capacity of 100 is not suitable as its state space size is tremendously large (as discussed in Section 3.3.1), which would cause the computational run time of the exact DP methods to be prohibitively high. So, instead, a smaller-scale ARM MDP problem with 2 fare classes (one high fare class and low fare class, i.e., \( i \in \{H, L\} \)), a flight capacity of 30 and a booking period of 10 time steps was considered for this purpose. The state space size is now of the order of \( 10^5 \) and the action space size is 2 with one action for opening the low fare class \((a = 1)\) and another for closing it \((a = 0)\). The fares of the high and low fare classes are specified to be $300 and $100 respectively. The mean number of passenger arrivals of the low and high fare class are 30 and 10 respectively and the passenger arrivals were modeled as a NHPP like before. The fare class booking cancellations were not considered in problem, so the
Cancellation rates were set to 0. All other problem parameters were kept unchanged.

Exact DP methods require the transition function and reward function of the MDP as inputs. For the given problem, the transition function is given by

$$T(s,a,s') = \begin{cases} 
P(\Delta b_{t,H})P(\Delta b_{t,L}), & \text{if } t' - t = 1 \\ 1, & \text{if } t' = 10, t = 0, b_H = 0 \text{ and } b_L = 0 (s = \text{initial state}) \\ 0, & \text{otherwise} \end{cases}$$ (3.16)

where $s = (b_H, b_L, t)$, $s' = (b'_H, b'_L, t')$, $\Delta b_{t,i} = b'_i - b_i$, and $P(\Delta b_{t,i})$ is the probability of fare class $i$ bookings changing by $\Delta b_{t,i}$ at time step $t$. As there are no booking cancellations, $\Delta b_{t,i} \geq 0$ at all time steps. When the fare classes are open, the probability $P(\Delta b_{t,i})$ is equivalent to the arrival probability of $\Delta b_{t,i}$ bookings given by the Poisson distribution, and, when the low fare class is closed, the probability of a nonzero low fare class bookings is zero, as shown below:

$$P(\Delta b_{t,H}) = \frac{\lambda_H(t)\Delta b_{t,H}e^{-\lambda_H(t)}}{\Delta b_{t,H}!}$$ (3.17)

$$P(\Delta b_{t,L}) = \begin{cases} 
\frac{\lambda_L(t)\Delta b_{t,L}e^{-\lambda_L(t)}}{\Delta b_{t,L}!}, & \text{if } a = 1 \\
1, & \text{if } a = 0, \Delta b_{t,L} = 0 \\
0, & \text{otherwise} \end{cases}$$ (3.18)

The reward function for this two-fare class ARM problem is the same as for the three fare class ARM problem, but without the terms for the middle fare class and with a reduced booking period. The reward function is now specified as:

$$R(s,s') = \begin{cases} 
\sum_{i \in I} \Delta b_{t,i} f_i, & \text{for } 0 \leq t \leq 9 \\
-\eta_L f_L - \eta_H f_H, & \text{if } t = 10 
\end{cases}$$ (3.19)
Figure 3.9: Optimal revenue percentage generated by agent during training for the ARM MDP with 2 fare classes.

Figure 3.10: Load factor achieved by agent during training for the ARM MDP with 2 fare classes.
To determine the optimal policy, the value iteration algorithm was used in this study. The DRL agent was trained to solve this MDP using deep Q-learning with the same hyperparameter setting as before.

The training plots are given in Figs. 4.10 and 4.11. As training progresses, the optimal revenue percentage generated by the agent can be seen to increase, while the load factor accomplished by the agent can be seen to decrease, until both of these performance metrics settle to values close to the $\mu_{RP}^*$ and $\mu_{LF}$ achieved by following the optimal policy respectively. The $\mu_{RP}^*$ and $\mu_{LF}$ obtained by following the optimal policy were 93.24% and 100.18% respectively. Note that the $\mu_{LF}$ of the optimal agent is slightly above 100% as expected for the reasons mentioned in Section 3.5.3. The difference in their $\mu_{RP}^*$ and $\mu_{LF}$ values are only 0.17% and 1.35% in magnitude respectively. These results demonstrate that the DRL agent, given proper training, can learn the optimal policy of the ARM seat inventory control MDP.

3.6 Summary

Solving the seat inventory control problem of ARM is critical for airlines to achieve financial success. In this study, we have formulated this leg-based ARM problem as a MDP and applied DRL to find the optimal seat inventory policy for airline revenue management with overbooking. Multiple fare classes with stochastic fare class demand, passengers arrivals and cancellations were considered in the problem. An air travel market simulator was developed based on the market dynamics and parameters for training and testing the agent. The simulation results demonstrate that the DRL agent can learn the optimal ARM policy through interactions with the market despite not having any prior knowledge of the market dynamics, passenger behavior and problem parameters. Also, the results show that the DRL agent generates close to optimal revenues in all flight episodes in all numerical experiments, outperforming the widely used traditional ARM heuristic rule-based decision method known as the EMSRb method. More specifically, the DRL agent achieved a 1.58% improvement in percentage of optimal revenues generated per flight and a 1.73% improvement in revenues generated.
per flight relative to the EMSRb method on average in all numerical experiments. The agent was
observed to achieve this by optimally controlling seat inventory in a way that protects seats for later
arriving higher class passengers from earlier arriving lower class passengers and overbook flights
before departure based on passenger arrival distributions and fare class cancellation rates respectively
such that the expected flight revenue is maximized and the load factor is near 100%. 
CHAPTER 4. DYNAMIC PRICING USING DEEP REINFORCEMENT LEARNING

Continuing along the lines of the work presented in the previous chapter, a DRL approach was used to address the problem of dynamic pricing for ARM. The discussion in this chapter covers sequentially the ARM dynamic pricing problem statement, our MDP formulation of the problem, the research and performance evaluation methodology, the implementation of the DRL solution method and the research findings from the different sets of numerical experiments carried out, and finally concludes with a summary of the work.

4.1 Problem Statement

In the leg-based seat inventory control problem considered in the previous chapter, the prices of the O-D fare products are varied indirectly by opening and closing fare classes at the start of each time step (or decision-making instance). In contrast, in dynamic pricing, the prices of the O-D fare products (or fare classes assuming that there is only one for each fare product) are varied directly by assigning a price to each fare product from an associated finite set of predetermined price points at each decision-making instance, as discussed previously in Chapter 1. The sale of a particular fare product may also be stopped to save seats for fulfilling the later arriving demand of higher fare products in a similar fashion to the seat inventory control problem. The objective of the problem considered in this study is to find the expected revenue-maximizing policy for dynamically pricing the fare products in each flight based on the arrival of passengers with different WTP at each time step, passenger cancellation behavior, time remaining till departure and remaining seat capacity.

The dynamic pricing problem considered here involves a single flight in which two types of fare
products are offered to the passengers: a low fare product (L) and a high fare product (H). The less
restricted fare product H is considered to be more expensive and targeted towards business passengers,
whereas the more restricted fare product L is cheaper and targeted towards leisure passengers. The set
of all fare products is denoted by \( J \), and, hence, \( J = \{ H, L \} \). The subscript \( j \) will be used as the index
of fare products, and, hence, \( j \in J \). If the two fare products are associated with the same O-D itinerary
and the origin and destination of the itinerary is the same as the origin and destination of the flight,
then the flight is associated with a single nonstop O-D market. Otherwise, the flight is associated with
multiple O-D markets, similar to the flight leg considered in the seat inventory control problem of
the previous chapter, in which case there needs to be a (coordination) policy for ensuring consistency
in the prices of the fare products in all of its associated flight legs in the leg-based approach (where
the dynamic pricing of flights legs are carried out separately). The set of price points associated
with the high and low products are \{ $400, $600 \} and \{ $150, $200 \} respectively. The chosen values
resemble the typical one-way O-D itinerary fares of a domestic flight in the US. Four different sets
of passengers with WTPs of $150 and $200 seeking the low fare product, and WTPs of $400 and
$600 seeking the high fare product are considered to arrive during the booking period of 182 days
(6 months). High fare product passengers may cancel their bookings free of cost. This implies that
the optimal dynamic pricing policy must also address flight overbooking optimally to maximize the
expected flight revenues. If a passenger gets denied boarding due to flight overbooking, the airline
reimburses the passenger in accordance to the denied boarding compensation policy adopted for the
seat inventory control problem, as discussed in Section 3.3.4. The passenger arrival and cancellation
distribution parameters are described in Sections 4.2.1 and 4.2.2 respectively below.

The capacity of the flight is again considered to be 100. As before, the booking window is split
into 182 days or time steps of equal duration. At the beginning of each time step, the airline can
select a price for each fare product from their corresponding set of price points for that time step,
and can also decide to close (stop offering) the low fare product for that time step. Passengers are
assumed to book a seat of their desired fare product only if it is open and if its fare is less than or
equal to their WTP. Before a flight departure, the airline can overbook the flight in anticipation of
future cancellations. In other words, the pre-departure number of bookings may exceed the physical flight capacity, but at departure, the number of booked passengers must be less than or equal to the flight capacity.

As the problem involves taking a series of actions at different points in time till the time of departure, this problem is a sequential decision-making problem. Taking the above problem specification into account, the dynamic pricing problem is suitable for being modeled as a MDP. In this work, we formulate this problem as a MDP and then find the optimal dynamic pricing policy by using a DRL approach. We conduct several numerical experiments to analyze the performance of our approach in terms of optimal revenue percentage generated relative to the EMSRb method and DP method.

### 4.2 Air Travel Market Simulator

The air travel market simulator developed previously for the seat inventory control problem was adapted for use in this problem to train and test the DRL and EMSRb agents. The simulator models the passenger arrivals and booking cancellations of different fare products at the different time steps of the booking period in the flight episodes based on the passenger arrival and cancellation models described in the following sections. This simulator is referred to as market simulator-A in the remainder of this chapter, where the ‘A’ stands for actual market. The set of all flight episodes generated by simulator-A is divided into two sets, a training set and a test set.

The research and performance evaluation methodology followed in this study is illustrated in the flow chart given in Fig. 4.1. Unlike before, for the sake of a more fair comparison between the two agents, the training set is not directly used to train the DRL agent. Instead, it serves as “historical” data which has been used to estimate the arrival and cancellation distribution parameters. The use of historical data to estimate passenger arrival, choice and cancellations models is the estimation approach commonly used in practice by airlines. As mentioned previously in Chapter 1, the historical data used by airlines in real-life are typically sourced from the airline’s own in-house booking data, market research firms and airline data providers such as IATA, ARC, GDSs, etc., and publicly
available data sources such as US DOT. These distribution parameter estimates are then provided to both the DRL and EMSRb agents for learning the initial optimal policy based on these estimates. For training the DRL agent using these estimated distribution parameters, another simulator, called simulator-T (training simulator), was developed based on these estimates. The underlying mechanics of the passenger arrival and cancellation models in both simulator-A and simulator-T is the same, but simulator-A uses the real market distribution parameters as specified by the problem while simulator-T uses the estimated market distribution parameters. The same estimated parameters are provided to the EMSRb agent which form the initial basis for its computations. The testing of both agents is carried out in simulator-A. During testing, both agents are allowed to update its estimates based on their observations.

Following the problem statement, the simulated passengers are allowed to book seats at most 182 days prior to the flight departure. Upon arriving, passengers are assumed to either book a seat if their desired fare product is open for sale and its price is no higher than their WTP or leave the booking system otherwise. High fare product passengers have the option of canceling their bookings at any time step following that in which they made the booking. Based on the knowledge of passenger booking arrivals and cancellations, the simulator computes the optimal revenue that may be generated in each flight episode. The revenue performance of the agents are assessed relative to the optimal flight revenues. The passenger arrival and cancellation models and the computation of the optimal revenue are described in the following sections.

4.2.1 Passenger Arrival Model

The arrival of fare product passengers with different WTPs has been modeled as a NHPP, for the same reasons discussed in Section 3.2.1. The time-varying mean arrival rate of the passengers with different WTP are determined using

$$
\lambda_k(t) = \int_t^{t+1} \lambda_k'(\tau; \hat{\alpha}_k, \hat{\beta}_k) \hat{A}_k d\tau,
$$

(4.1)
“Actual” market simulator-A

Generate historical flight episode data

Estimate market distribution parameters

Build “training” simulator-T based on these market parameter estimates and train DRL agent

Set EMSRb agent’s model parameters based on these market parameter estimates

Test DRL agent on (actual) market simulator-A

Test EMSRb agent on (actual) market simulator-A

Compare results

Simulate market perturbations in simulator-A and test agents’ adaptability and performance

Figure 4.1: The research and performance evaluation methodology for the DRL and EMSRb agents employed for dynamic pricing of flights
where $\lambda_k(t)$ is the mean arrival rate of passenger group with WTP $k$ ($k \in \{150, 200, 400, 600\}$) at time step $t$. $\lambda'_k(\tau; \tilde{\alpha}_k, \tilde{\beta}_k)$ is the probability density function value of a beta distribution with shape parameters $\tilde{\alpha}_k$ and $\tilde{\beta}_k$ corresponding to passenger group with WTP $k$ at time $\tau$, and $\hat{A}_k$ is the mean number of arrivals of passenger group with WTP $k$ in each flight episode. The probability density function of the beta distribution associated with the different WTP passenger groups is defined as

$$
\lambda'_k(\tau) = \frac{(\tau \hat{\alpha}_k - 1)(1 - \tau \hat{\beta}_k - 1)}{B(\hat{\alpha}_k, \hat{\beta}_k)}, \tag{4.2}
$$

where

$$
B(\hat{\alpha}_k, \hat{\beta}_k) = \frac{\Gamma(\hat{\alpha}_k)\Gamma(\hat{\beta}_k)}{\Gamma(\hat{\alpha}_k + \hat{\beta}_k)} \tag{4.3}
$$

and $\Gamma$ is the Gamma function. During simulation, the number of arrivals of WTP $k$ passenger group at each time step $t$ ($A_{t,k}$) is determined by sampling from a Poisson distribution with the mean arrival rate $\lambda_k(t)$ (i.e., $A_{t,k} \sim \text{Poisson}(\lambda_k(t))$). The mean number of arrivals of passengers with WTP $150$, $200$, $400$, and $600$ were set to 70, 40, 21 and 14 respectively. So, the average number of total passenger arrivals in each flight was 145. Keeping the ratio of average total demand to flight capacity as 1.45 ensures that the agent deals with overbooking in most flight episodes. The shape parameters $\tilde{\alpha}_k$ and $\tilde{\beta}_k$ of the beta distributions determine the passenger arrival patterns of the passenger groups with different WTPs. Table 4.1 lists the shape parameter values used in simulator-A.

**Table 4.1: The beta distribution shape parameter values of the WTP passenger groups**

<table>
<thead>
<tr>
<th>WTP ($k$)</th>
<th>$\tilde{\alpha}_k$</th>
<th>$\tilde{\beta}_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$150$</td>
<td>28.35</td>
<td>1.65</td>
</tr>
<tr>
<td>$200$</td>
<td>15.4</td>
<td>4.6</td>
</tr>
<tr>
<td>$400$</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>$600$</td>
<td>12</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 4.2 shows how the arrival rate of each WTP passenger group varies with time throughout the booking period. A number of real-world passenger arrival characteristics are evident from the figure: 1) most lower WTP passengers have a greater likelihood of arriving earlier than higher WTP
passengers; 2) the order of arrivals is uncertain; 3) the number of passenger arrivals of each passenger group in each flight episode is stochastic; and 4) the mix of passenger arrivals of the WTP passenger groups is uncertain.

Figure 4.2: Plot of passenger arrivals rates of different WTP passenger groups

4.2.2 Cancellation Model

The cancellation model is the same as the one used for the seat inventory control study. For the high fare product passengers \((k \in \{400, 600\})\), the passenger cancellation rate is considered to be 0.01% at all time steps preceding the last two time steps. In the last two time steps \((t \in \{180, 181\})\), the cancellation rate of high fare product passengers is considered to be 5%. As low fare product passengers \((k \in \{150, 200\})\) do not have the option of cancelling their bookings, their cancellation rate is set to 0% at all time steps.

4.2.3 Optimal Revenue Computation

Given the knowledge of the number of passenger arrivals, their WTP, and their future cancellation decision (commitment to fly) at each time step of a given flight episode, the upper bound of flight revenue that can be generated from booking passengers in that flight can be found by solving the
integer linear program given in Eq. 4.4, where the objective function is total flight revenue, \( n_k' \) is the number of arrivals of passenger with WTP \( k \) at time step \( t \), \( p_t' \) is a binary decision variable indicating whether price point \( l \) \( (l \in L, L = \{150, 200, 400, 600\}) \) is selected or not for its corresponding fare product at time step \( t \), \( B \) is the bumping cost factor as before, \( \eta_l \) is the number of passengers bumped at the end of the booking period who paid fare \( l \), and \( b_{150}, b_{200}, b_{400} \) and \( b_{600} \) are the total number of bookings made at prices \( $600, $400, $200 \) and \( $150 \) respectively in the flight.

\[
\begin{align*}
\max \quad & \sum_{t=0}^{181} \{600n_{600}'p_{600}' + 400(n_{600}' + n_{400}')(1 - p_{600}') \\
& + 200n_{200}'p_{200}' + 150(n_{200}' + n_{150}')p_{150}' \} \\
& - 150B\eta_{150} - 200B\eta_{200} - 675\eta_{400} - 675\eta_{600} \\
\text{s.t.} \quad & p_{200}' + p_{150}' \leq 1 \quad \forall t \in \{0, \ldots, 181\} \\
& n^B \geq n^B - 100 \\
& \eta^T \geq \sum_{t=0}^{181} \{n_{600}'p_{600}' + (n_{600}' + n_{400}') (1 - p_{600}') \\
& + n_{200}'p_{200}' + (n_{200}' + n_{150}')p_{150}' \} \\
& \eta_{150} \leq \eta^T \\
& \eta_l \leq b_l \quad \forall l \in L \\
& b_{150} = \sum_{t=0}^{181} n_{150}'p_{150}' \\
& b_{200} = \sum_{t=0}^{181} n_{200}'p_{200}' + n_{200}'p_{150}' \\
& b_{400} = \sum_{t=0}^{181} -n_{400}'p_{600}' + n_{400}' \\
& b_{600} = \sum_{t=0}^{181} n_{600}' \\
& \eta^T, \eta_l \geq 0 \quad \forall l \in L \\
& p_t' \in \{0, 1\} \quad \forall t \in \{0, \ldots, 181\}, \forall l \in \{150, 200, 600\}
\end{align*}
\]

The total flight revenue is equal to the revenue generated from bookings minus the bumping cost. In accordance with the denied boarding compensation policy specified in the problem, the
total bumping cost amounts to \(-150B\eta_{150} - 200B\eta_{200} - 675\eta_{400} - 675\eta_{600}\). The first constraint specifies that at most one price point can be selected for the low fare product at any given time step. When both \(p'_{150}\) and \(p'_{200}\) are set equal to zero, the low fare product is not available for sale at time step \(t\). For the high fare product, there is only one price decision variable, \(p'_{600}\), to indicate its price and availability, as it is considered to be never closed for sale. The notation \(\eta^T\) denotes the total number of passengers that need to be bumped at flight departure and \(n^B\) the total number of passengers booked. The rest of the constraints ensure that \(\eta^T = \max(n^B - 100, 0)\) and specify the number of passengers to be bumped from each group who paid fare \(l\).

This mathematical program was solved for each flight episode with its passenger arrival and cancellation data to determine its optimal revenue, which is used to benchmark the performance of the agents. More specifically, the agents’ performance is measured in terms of the average percentage of optimal revenue it is able to generate in the flight episodes at the end of training in simulator-T and during testing in simulator-A.

4.3 Markov Decision Process Formulation

To learn the optimal policy of the ARM dynamic pricing problem, it is first formulated as a MDP. The components of the MDP formulation are described in the following sections.

4.3.1 States

A state \(s\) in the state space \(S (s \in S)\) of our MDP is represented by a vector of integer state variables to capture the booking and cancellation counts at the various price points and the current time information. A state \(s\) is defined as

\[
s = (b_{600}, b_{400}, b_{200}, b_{150}, c_{600}, c_{400}, t),
\]

which includes four booking count variables \(b_{600}, b_{400}, b_{150}\), and \(b_{100}\) that keep track of the number of bookings made at prices $600, $400, $200 and $150 respectively, two cancellation count variables
\( c_{600} \) and \( c_{400} \) that keep track of the number of cancellations of bookings with prices $600 and $400 respectively, and the time step variable \( t \) to keep track of the current time step in the booking period. Note that, at any state \( s \), the total number of high fare product bookings is equal to \( b_{600} + b_{400} \) and the total number of low fare product bookings is equal to \( b_{200} + b_{150} \). As an example, a state \( s = (5, 10, 35, 40, 1, 2, 180) \) denotes that the current time step is 180, the present number of bookings made at prices $600, $400, $200 and $150 are 5, 10, 35 and 40 respectively, and the present number of cancellations of bookings with prices $600 and $400 are 1 and 2 respectively. At the beginning of each flight episode, the initial state is \( s = (0, 0, 0, 0, 0, 0, 0) \). The representation of the states in this way results in a state space of size in the order of higher than approximately \( 10^{10} \) if the Poisson arrival distributions are truncated at the points beyond which the arrival probabilities become less than the order of \( 10^{-2} \).

4.3.2 Actions

At the beginning of each time step, the agent can take an action \( a \) to specify the prices of fare products, \( f_H \) and \( f_L \), from their corresponding set of price points and the availability of the low fare product during the current time step. The actions constitute the action space \( A (a \in A) \). The set of allowed actions for the agent and their interpretations is listed in Table 4.2. These actions allow the agent to vary the prices of the fare products and protect seats for high fare product passengers based on the remaining capacity and time to departure. In total, the agent has to take 182 actions in each flight episode before flight departure.

Table 4.2: Actions the agent can take to control the prices of the fare products and the availability of the low fare product

<table>
<thead>
<tr>
<th>Action</th>
<th>( f_H )</th>
<th>( f_L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tilde{a}_1 )</td>
<td>$400</td>
<td>$150</td>
</tr>
<tr>
<td>( \tilde{a}_2 )</td>
<td>$400</td>
<td>$200</td>
</tr>
<tr>
<td>( \tilde{a}_3 )</td>
<td>$400</td>
<td>close</td>
</tr>
<tr>
<td>( \tilde{a}_4 )</td>
<td>$600</td>
<td>$150</td>
</tr>
<tr>
<td>( \tilde{a}_5 )</td>
<td>$600</td>
<td>$200</td>
</tr>
<tr>
<td>( \tilde{a}_6 )</td>
<td>$600</td>
<td>close</td>
</tr>
</tbody>
</table>
4.3.3 Model Dynamics

After each time step, the state variable \( t \) is incremented by one as time progresses chronologically. During each time step, passengers with different WTPs arrive looking to book their desired fare products. If their desired fare product is open for sale and its price is less than or equal to their WTP, then the passenger is considered to make a booking and the corresponding booking count variable is incremented. Also, some previously booked high fare product passengers may cancel their bookings during each time step. The corresponding cancellation count variables are incremented and the corresponding booking count variables are decremented accordingly to reflect these cancellations. So, the net change in the booking and cancellation count variables associated with price point \( l \) in time step \( t \) are

\[
\Delta b_{t,l} = \beta_{t,l} - C_{t,l}, \tag{4.6}
\]

and

\[
\Delta c_{t,l} = C_{t,l} \tag{4.7}
\]

respectively, where \( \beta_{t,l} \) and \( C_{t,l} \) are the total number of new bookings and booking cancellations associated with price point \( l \) in the given time step \( t \) respectively. Let \( \alpha_{t,l} \) denote the number of passenger arrivals with WTP \( l \) in time step \( t \). Based on the passenger booking behavior described above,

\[
\beta_{t,l} = \begin{cases} 
\alpha_{t,l}, & \text{if } l = 600, f_H = l \text{ or } l = 200, f_L = l \\
\alpha_{t,l} + \alpha_{t,l'}, & \text{if } l = 400, l' = 600, f_H = l \text{ or } l = 150, l' = 200, f_L = l \\
0, & \text{otherwise}
\end{cases} \tag{4.8}
\]

The values of \( \alpha_{t,l} \) and \( C_{t,l} \) are generated by the simulators’ arrival and cancellation models respectively. Note that \( C_{t,l} = 0 \) for \( l \in 150, 200 \) as the low fare product cancellation rate is 0. Once the terminal state (\( t = 182 \)) is reached, all actions will lead to the initial state.
4.3.4 Reward Function and Discount Factor

The reward function \( R(s, s') \) specifies the revenue received by the agent at the end of each time step \( t \) when it moves from one state \( s \) to another state \( s' \). It is defined as:

\[
R(s, s') = \begin{cases} 
\sum_{l \in L} \Delta b_{t,l}, & \text{for } 0 \leq t \leq 181 \\
- \sum_{l \in \{150, 200\}} \eta_l B_l - 675 \eta_{400} - 675 \eta_{600}, & \text{if } t = 182,
\end{cases}
\]

(4.9)

where \( s = (b_{600}, b_{400}, b_{200}, b_{150}, c_{600}, c_{400}, t) \), \( s' = (b'_{600}, b'_{400}, b'_{200}, b'_{150}, c'_{600}, c'_{400}, t') \) and \( \Delta b_{t,l} = b'_l - b_l \). At all time steps before flight departure (0 ≤ \( t \) ≤ 181), the revenue generated from bookings associated with price point \( l \) is \( \Delta b_{t,l} \), the total amount resulting from adding up all the fares collected from the bookings and subtracting all the fare reimbursements due to the booking cancellations. Note that at most one price point is selected for each fare product at any given time step \( t \). So, if a price point \( l \) is not selected at time step \( t \) or the number of cancellations of bookings associated with price point \( l \) is more than the number of bookings in time step \( t \), \( \Delta b_{t,l} \leq 0 \). At the terminal state (\( t = 182 \)), if the total number of passenger bookings is greater than the flight capacity, the agent will incur a bumping cost for having to deny boarding to some passengers. This bumping cost is equal to

\[
- \sum_{l \in \{150, 200\}} \eta_l B_l - 675 \eta_{400} - 675 \eta_{600}.
\]

The total number of passengers that need to be bumped (\( \eta^T \)) to ensure that the flight load factor is less than or equal to one can be determined using

\[
\eta^T = \max \left( \sum_{l \in L} b_l - \kappa, 0 \right) \text{ at } t = 182.
\]

(4.10)

Passengers are assumed to be bumped in ascending order of the fare they paid for their booking. In other words, passenger bumping starts with the bookings associated with the lowest price point and keeps moving onto higher price points until the remaining number of booked passengers is equal to the flight capacity.
The number of passengers to be bumped from \( b_{150}, b_{200}, b_{400} \) and \( b_{600} \) at \( t = 182 \) can be determined using the following set of equations:

\[
\eta_{150} = \begin{cases} 
\eta^T, & \text{if } \eta^T \leq b_{150} \\
 b_{150}, & \text{if } \eta^T > b_{150}
\end{cases} \tag{4.11}
\]

\[
\eta_{200} = \begin{cases} 
\eta^T - \eta_{150}, & \text{if } \eta^T - \eta_{150} \leq b_{200} \\
 b_{200}, & \text{if } \eta^T - \eta_{150} > b_{200}
\end{cases} \tag{4.12}
\]

\[
\eta_{400} = \begin{cases} 
\eta^T - \eta_{150} - \eta_{200}, & \text{if } \eta^T - \eta_{150} - \eta_{200} \leq b_{400} \\
 b_{400}, & \text{if } \eta^T - \eta_{150} - \eta_{200} > b_{200}
\end{cases} \tag{4.13}
\]

\[
\eta_{600} = \eta^T - \eta_{150} - \eta_{200} - \eta_{400} \tag{4.14}
\]

Assuming the effect of monetary discounting within the booking period to be insignificant, a discount factor of 0.9999 was chosen to facilitate convergence during training.

### 4.4 Solution Method

To solve the MDP formulated in the preceding section and learn and evaluate the optimal dynamic pricing policy of market simulator-A, a DRL agent is first trained in simulator-T and then tested in simulator-A, following the research methodology described in Section 4.2. For either simulator, the interactions taking place between the agent and the market simulator is visualized in Fig. 4.3, where \( s, a \) and \( r \) are the present state, action and reward respectively, and \( s' \) and \( r' \) are the state and reward at the next immediate time step. The performance of the DRL agent is compared to that of the EMSRb agent during training and testing. The configuration of both the DRL and EMSRb agents are discussed in the following sections.
4.4.1 DRL Agent

For determining the learning algorithm and tuning the neural network configuration and the values of the most significant hyperparameters of the learning algorithm, a series of numerical experiments were conducted in simulator-T by varying the learning algorithm, number of hidden layers, number of hidden neurons, target model update interval (TMU) and learning rate (LR). In each experiment, the post-training revenue performance and load factor of the different DRL agents with distinct hyperparameter settings were recorded. The three different variations of DQL learning algorithm tested are: (regular) DQL, deep double Q-learning (DDQL) and DQL with dueling DQN (DQL-dueling). The results of the experiments carried out with DQL is given in Table 4.3, where, among all the hyperparameter settings tried out, experiment number 24 produced the highest $\mu_{RP^*}$ with the second lowest $\sigma_{RP^*}$. Keeping this best hyperparameter setting fixed, the other two learning algorithms were then tested and found to produce similar results with a slightly lower $\mu_{RP^*}$, as listed in Table 4.4. Based on these two sets of experimental results, the learning algorithm DQL along with the hyperparameter setting of experiment number 24 were chosen for the rest of the experiments in this study.
**Table 4.3: Results of neural network configuration and hyperparameter tuning experiments**

<table>
<thead>
<tr>
<th>Exp. no.</th>
<th>No. hidden layers</th>
<th>No. hidden neurons</th>
<th>TMU</th>
<th>LR</th>
<th>$\mu_{RP^*}$ (%)</th>
<th>$\sigma_{RP^*}$ (%)</th>
<th>$\mu_{LF}$ (%)</th>
<th>$\sigma_{LF}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>64</td>
<td>0.01</td>
<td>0.001</td>
<td>93.66</td>
<td>2.92</td>
<td>99.31</td>
<td>5.85</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>64</td>
<td>0.01</td>
<td>0.0001</td>
<td>91.73</td>
<td>4.83</td>
<td>97.38</td>
<td>7.96</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>64</td>
<td>10000</td>
<td>0.001</td>
<td>94.05</td>
<td>3.16</td>
<td>100.7</td>
<td>8.27</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>64</td>
<td>10000</td>
<td>0.0001</td>
<td>84.07</td>
<td>5.05</td>
<td>118.11</td>
<td>10.6</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>128</td>
<td>0.01</td>
<td>0.001</td>
<td>93.81</td>
<td>2.9</td>
<td>100.05</td>
<td>7.42</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>128</td>
<td>0.01</td>
<td>0.0001</td>
<td>93.99</td>
<td>3.22</td>
<td>101.36</td>
<td>7.98</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>128</td>
<td>10000</td>
<td>0.001</td>
<td>92.83</td>
<td>3.9</td>
<td>93.27</td>
<td>7.54</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>128</td>
<td>10000</td>
<td>0.0001</td>
<td>86.63</td>
<td>3.06</td>
<td>88.18</td>
<td>4.97</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>256</td>
<td>0.01</td>
<td>0.001</td>
<td>93.53</td>
<td>4.22</td>
<td>98.44</td>
<td>7.03</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>256</td>
<td>0.01</td>
<td>0.0001</td>
<td>92.58</td>
<td>3.56</td>
<td>99.29</td>
<td>8.61</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>256</td>
<td>10000</td>
<td>0.001</td>
<td>93.34</td>
<td>2.7</td>
<td>99.74</td>
<td>6.05</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>256</td>
<td>10000</td>
<td>0.0001</td>
<td>94.05</td>
<td>2.52</td>
<td>99.31</td>
<td>5.47</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>64</td>
<td>0.01</td>
<td>0.001</td>
<td>92.58</td>
<td>3.33</td>
<td>103</td>
<td>6.02</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>64</td>
<td>0.01</td>
<td>0.0001</td>
<td>90.37</td>
<td>3.88</td>
<td>97.23</td>
<td>7.95</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>64</td>
<td>10000</td>
<td>0.001</td>
<td>93.9</td>
<td>2.35</td>
<td>100.07</td>
<td>5.56</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>64</td>
<td>10000</td>
<td>0.0001</td>
<td>91.69</td>
<td>2.82</td>
<td>106.5</td>
<td>5.74</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>128</td>
<td>0.01</td>
<td>0.001</td>
<td>93.85</td>
<td>2.91</td>
<td>103.45</td>
<td>7.44</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>128</td>
<td>0.01</td>
<td>0.0001</td>
<td>92.84</td>
<td>3.27</td>
<td>105.4</td>
<td>7.43</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>128</td>
<td>10000</td>
<td>0.001</td>
<td>94.28</td>
<td>2.98</td>
<td>98.68</td>
<td>7.44</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>128</td>
<td>10000</td>
<td>0.0001</td>
<td>93.22</td>
<td>3.16</td>
<td>97.63</td>
<td>6.72</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>256</td>
<td>0.01</td>
<td>0.001</td>
<td>93.57</td>
<td>3.33</td>
<td>97.04</td>
<td>7.27</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>256</td>
<td>0.01</td>
<td>0.0001</td>
<td>93.99</td>
<td>3.31</td>
<td>101.45</td>
<td>8.28</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>256</td>
<td>10000</td>
<td>0.001</td>
<td>93.09</td>
<td>3.3</td>
<td>100.06</td>
<td>7.17</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>256</td>
<td>10000</td>
<td>0.0001</td>
<td>94.51</td>
<td>2.37</td>
<td>100.31</td>
<td>5.41</td>
</tr>
</tbody>
</table>

**Table 4.4: Results of different learning algorithms with the best found hyperparameter setting**

<table>
<thead>
<tr>
<th>Exp no.</th>
<th>Learning algorithm</th>
<th>Hyperparameter setting</th>
<th>$\mu_{RP^*}$ (%)</th>
<th>$\sigma_{RP^*}$ (%)</th>
<th>$\mu_{LF}$ (%)</th>
<th>$\sigma_{LF}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>DQL</td>
<td>Exp. 24</td>
<td>94.51</td>
<td>2.37</td>
<td>100.31</td>
<td>5.41</td>
</tr>
<tr>
<td>25</td>
<td>DDQL</td>
<td>Exp. 24</td>
<td>93.51</td>
<td>3.74</td>
<td>96.34</td>
<td>8.05</td>
</tr>
<tr>
<td>26</td>
<td>DQL-dueling</td>
<td>Exp. 24</td>
<td>94.42</td>
<td>2.31</td>
<td>100.04</td>
<td>5.14</td>
</tr>
</tbody>
</table>
4.4.1.1 Neural Network Architecture

Based on the results of the hyperparameter tuning experiments, a dense, feedforward neural network with three hidden layers was chosen to approximate the Q-values. The input layer has 8 nodes, one node for each variable in the state vector and an additional bias node. Each of the three hidden layers are composed of 256 ReLU activated hidden neurons. The output layer has 6 linearly activated nodes, one node for each action. The architecture of the neural network is depicted in Fig. 4.4.

4.4.1.2 Hyperparameter Setting

During training and testing, the deep Q-learning algorithm updated the weights of the neural network based on the agent-market simulator interaction samples. Based on the results of the hyperparameter tuning experiments, a hard weight update approach was chosen by setting the target model update interval parameter of DQL to 10000. The learning rate of the optimizer was chosen to be 0.0001. In all the hyperparameter tuning experiments, the optimization algorithm used was Adam;
the training batch size was 32; and a linear annealed $\epsilon$-greedy policy was chosen as the exploration policy, where the value of $\epsilon$ was linearly decreased from 1 to 0 over 9500 flight episodes and kept at 0 for the last 500 episodes of the training phase. These settings were kept unchanged for the rest of the study.

4.4.2 EMSRb Agent

The EMSRb agent uses the market distribution estimates to carry out its computations based on the theory and equations described in Chapter 2. Each price point acts as a fare class for which the EMSRb agent computes a booking limit. At a given time step, if the booking limit associated with a price point is reached or if it is zero, then the price point is not selected (closed) and instead the next higher price point is selected (opened). As the EMSRb agent is model-based, the two most important hyperparameters for the EMSRb agent are the “model update interval” (MUI) and the “observation count” (OC) for updating the estimates. The estimates are updated periodically after every $n$ episodes, where $n$ is equal to the model update interval, and the updated estimates are based on the average of last $m$ observations, where $m$ is equal to the specified observation count for updating.

A series of hyperparameter tuning experiments were carried out for the EMSRb agent in simulator-T as was done for the DRL agent. The results obtained for the different combinations of model update interval and observation count are listed in Table 4.5. The hyperparameter setting of experiment number 4 is chosen for the EMSRb agent as it was found to generate the highest $\mu_{RP}$ with a low $\sigma_{RP}$.

4.5 Results

Following the research methodology outlined in Fig. ??, the DRL agent was first trained in simulator-T and the market parameter estimates were provided as inputs to the EMSRb agent, after which both the agents were tested separately in simulator-A. To test the agents’ adaptability to changes in the market, market perturbations were introduced in simulator-A and their performance loss and recovery were recorded. In all experiments, the key performance metrics were the average
Table 4.5: Results of EMSRb agent’s hyperparameter tuning experiments

<table>
<thead>
<tr>
<th>Exp. no.</th>
<th>MUI</th>
<th>OC</th>
<th>$\mu_{RP}^*$ (%)</th>
<th>$\sigma_{RP}^*$ (%)</th>
<th>$\mu_{LF}$ (%)</th>
<th>$\sigma_{LF}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>200</td>
<td>89.41</td>
<td>3.72</td>
<td>98.57</td>
<td>4.87</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>350</td>
<td>89.99</td>
<td>3.58</td>
<td>99.29</td>
<td>4.63</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>500</td>
<td>90.20</td>
<td>3.47</td>
<td>99.53</td>
<td>4.52</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1000</td>
<td>90.33</td>
<td>3.55</td>
<td>99.73</td>
<td>4.55</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>200</td>
<td>89.43</td>
<td>3.71</td>
<td>98.58</td>
<td>4.87</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>350</td>
<td>89.96</td>
<td>3.51</td>
<td>99.26</td>
<td>4.63</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>500</td>
<td>90.15</td>
<td>3.47</td>
<td>99.49</td>
<td>4.51</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>1000</td>
<td>90.31</td>
<td>3.56</td>
<td>99.68</td>
<td>4.54</td>
</tr>
<tr>
<td>9</td>
<td>50</td>
<td>200</td>
<td>89.33</td>
<td>3.77</td>
<td>98.42</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>350</td>
<td>89.86</td>
<td>3.62</td>
<td>99.11</td>
<td>4.78</td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>500</td>
<td>90.04</td>
<td>3.54</td>
<td>99.34</td>
<td>4.68</td>
</tr>
<tr>
<td>12</td>
<td>50</td>
<td>1000</td>
<td>90.12</td>
<td>3.58</td>
<td>99.51</td>
<td>4.71</td>
</tr>
</tbody>
</table>

percentage of optimal revenue and load factor the agents achieved by following their respective policies. The results of training, testing and adaptability experiments are presented in the following sections.

4.5.1 Training in Simulator-T

The training process followed for the seat inventory control problem is followed here as well. While training the DRL agent, the weights of the agent’s neural network model were saved at intervals of 250 flight episodes. So, for 10000 training flight episodes, a total of 40 neural network models were saved. Each of these models were tested on a separate unseen set of 300 test flight episodes generated by simulator-T. During testing in simulator-T, the neural network weights are not updated and the models follow a greedy policy. Out of all saved models, the model that gave the best performance in terms of the average percentage of optimal revenue and the average load factor was selected as the final model. Training was considered to converge when the change in optimal revenue percentage moving average is less than 0.01 for 10 consecutive flight episodes.

This same process was followed for the hyperparameter tuning experiments. The training plots of the DRL agent with our chosen hyperparameter setting are presented in Figs. 4.5 and 4.6. A trend
similar to that observed for the training plots of the seat inventory control problem can be seen here. With accumulation of experience during training, the agent’s average optimal revenue percentage increases to 94.51% and average load decreases to 100.31%.

4.5.2 Testing in Simulator-A

After training of the DRL agent in simulator-T was completed, both the DRL and EMSRb agents were tested in simulator-A. During testing, both the DRL and the EMSRb agents were allowed to continue updating its neural network weights and parameter estimates respectively based on the observations from the market. The test performance plots are given in Figs. 4.7 and 4.8. The overall average optimal revenue percentages of the DRL and EMSRb agents were found to be 93.08% and 89.82% respectively. As evident from Fig. 4.7, the DRL agent consistently scores a higher optimal revenue percentage in each episode than the EMSRb agent, leading by 3.26% on average. Although the overall average load factors of the DRL and EMSRb agents of 99.32% and 98.63% respectively are similar, the difference in revenue performances clearly suggest that the DRL agent is following a superior policy.

The DRL agent’s price evolution graph and passenger arrival bar plot of test flight episode 166, shown in Fig. 4.9, reveals that the agent has learned to dynamically price the fare products in a way that exploits the WTP of the arriving passengers and protects seats for later arriving passengers with higher WTP to maximize the total expected flight revenue. In the first part of the flight episode, when there are typically no passenger arrivals due to infinitesimally small probabilities in the passenger arrival distributions, the agent is indifferent between the choices of price points for the fare products. Between time steps 50 and 75, most of the low fare product passengers with WTP $150 arrive, during which the agent sets the price of the low fare product to $150. After around time step 80, the agent increases the low fare product’s price to $200 and keeps it fixed there until around time step 145 to book the passengers with WTP $200 and protect seats for later arriving higher WTP passengers from passengers with WTP $150. As the flight capacity is getting filled up, the agent then keeps the low fare product closed for the next around 20 time steps, as indicated by the dotted orange line, to
Figure 4.5: Average optimal revenue percentage generated by DRL agent during training

Figure 4.6: Average load factor achieved by DRL agent during training
Figure 4.7: Optimal revenue percentage generated by agents during testing in simulator-A

Figure 4.8: Load factor achieved by agents during testing in simulator-A
Figure 4.9: The DRL agent’s price evolution graph (top) and passenger arrival bar plot (bottom) of test flight episode 166

protect seats for the high fare product passengers. The agent can be observed to increase the price of the high fare product at around time step 165 so that it is the same as the WTP of the arriving passengers in the last part of the episode.

4.5.3 DRL vs Dynamic Programming: Learning the Optimal Dynamic Pricing Policy

The optimality of the DRL agent’s dynamic pricing policy can be assessed by comparing the percentage of optimal revenue generated by following the DRL agent’s policy with that obtained by following the optimal policy of the ARM dynamic pricing MDP, as explained in Chapter 3. As mentioned previously, the optimal policy of the ARM MDP can be derived using DP methods. However, DP methods are not attractive for use in practice as they require the complete knowledge of
the transition function, which is never known exactly in real life, and their computational expense is prohibitively high for realistically scaled ARM problems. Following the approach used for this test for the seat inventory control problem, the current ARM MDP problem is scaled down by reducing the flight capacity to 10, the booking period to 5 time steps, and the mean number of arriving passengers with WTP $150, $200, $400, and $600 to 2, 2, 3 and 3 respectively. All the arriving passengers with WTP $150, $200, $400, and $600 are now considered to arrive in the second, third, fourth and fifth time step respectively. The state space size is now of the order of $10^4$. The action space is the same as before. Booking cancellations were not considered in problem, so the cancellation rates were set to 0 and the cancellation state variables are removed from the state representation such that $s = (b_{600}, b_{400}, b_{200}, b_{150}, t)$. All other problem parameters were kept unchanged. For the given problem, the transition function can be computed using the following set of equations:

$$T(s, a, s') = \begin{cases} 
\prod_{l \in L} P(\Delta b_t, l) & \text{if } t' - t = 1 \\
1, & \text{if } t' = 5, \ s = \text{initial state} \\
0, & \text{otherwise}, \end{cases}$$

(4.15)

$$P(\Delta b_{t,l}) = \begin{cases} 
\hat{P}(\Delta b_{t,l}; \lambda_l(t)), & \text{if } l = 600, \ f_H = l \text{ or } l = 200, \ f_L = l \\
\sum_{q=0}^{\Delta b_{t,l}} \hat{P}(q; \lambda_l(t)) \hat{P}(\Delta b_{t,l} - q; \lambda_{l'}(t)), & \text{if } l = 400, \ f_H = l \text{ or } l = 200, \ f_L = l \\
1, & \text{if } \Delta b_{t,l} = 0, \ l \in \{150, 200\}, f_L = \text{close} \\
1, & \text{if } \Delta b_{t,l} = 0, \ l \in \{150, 200\}, f_H \neq l \\
1, & \text{if } \Delta b_{t,l} = 0, \ l \in \{400, 600\}, f_H \neq l \\
0, & \text{otherwise}, \end{cases}$$

(4.16)

$$\hat{P}(\Delta b; \lambda) = \frac{\lambda^{\Delta b} e^{-\lambda}}{\Delta b!},$$

(4.17)

where $l'$ is the price point immediately higher than $l$ and $\hat{P}(\Delta b; \lambda)$ is the Poisson probability mass function. The reward function is the same as the one given in Eq. 4.9. To determine the optimal
policy, the value iteration algorithm was used in this study. The DRL agent was trained to solve this MDP using deep Q-learning with the same hyperparameter setting as before.

The training plots are given in Figs. 4.10 and 4.11. As training progresses, the optimal revenue percentage generated by the agent can be seen to increase, while the load factor accomplished by the agent can be seen to decrease, until both of these performance metrics settle to values close to that achieved by following the optimal policy. More specifically, the $\mu_{RP}^*$ and $\mu_{LF}$ achieved by the DRL agent during testing were 96.67% and 100.33% respectively. The $\mu_{RP}^*$ and $\mu_{LF}$ obtained by following the optimal policy were 96.79% and 96.63% respectively. The difference in their $\mu_{RP}^*$ values is only 0.12%. These results demonstrate that the DRL agent, given proper training, can learn the optimal policy of the ARM dynamic pricing MDP without requiring as inputs the explicit transition function and reward function of the problem unlike its DP counterparts.

### 4.5.4 Adaptability Testing

Air travel markets are generally not stationary. The total demand, the demand ratio for the fare products, and passenger characteristics typically change with time due to changes in socioeconomic and demographic factors (such as population size, age of population, disposable income, standards of living, levels of education, etc.), competitors’ and alternative mode of transports’ offerings and prices, regulations, jet fuel cost, and infrastructure developments and a host of other factors. Also, the occurrence of unfavorable events like economic recession or slowdown, instability in financial markets, terror attacks and pandemics, such as the current COVID-19 one, drastically reduces the volume of O-D air travel market demands. So, it is highly desirable for an ARM system to be able to adapt its policy to changes in the market conditions.

To test the adaptability of the DRL and EMSRb agents, two different types of market perturbations were introduced in simulator-A in two separate experiments. In light of the current ongoing COVID-19 pandemic, in adaptability experiment-1, a negative high fare product demand shock was simulated by reducing the mean number of arriving passengers with WTP $600 from 14 to 5 and arriving passengers with WTP $400 from 21 to 10. A similar but more severe negative demand shock
Figure 4.10: Optimal revenue percentage generated by agent during training for the small scale dynamic pricing MDP

Figure 4.11: Load factor achieved by agent during training for the small scale dynamic pricing MDP
is currently being faced by the global airline industry due to the current COVID-19 pandemic. In adaptability experiment-2, a surge in passenger cancellations was simulated by increasing the cancellation rates of high fare product passengers from 0.01% to 5% at all time steps. As the adaptability of the DRL agent depends primarily on its exploration policy and that of the EMSRb agent on its market estimates updating policy, different hyperparameter settings for both agents were first investigated in the new market environment of experiment-1 to tune their hyperparameters for adaptability testing. The results of hyperparameter tuning experiments for the DRL and EMSRb agents are given in Tables 4.6 and 4.7 respectively, where NEE is the number of initial episodes in which exploration was carried out. For the DRL agent, the hyperparameter setting associated with experiment number 5, and for the EMSRb agent, the hyperparameter setting associated with experiment number 3, from their respective experiment sets produced the highest $\mu_{RP^*}$. These best respective set of hyperparameters were used for both agents in both experiments.

Table 4.6: Results of DRL agent’s hyperparameter tuning experiments for adaptability test

<table>
<thead>
<tr>
<th>Exp. no.</th>
<th>NEE</th>
<th>TMU</th>
<th>LR</th>
<th>Lowest avg. RP* (%)</th>
<th>Highest avg. RP* (%)</th>
<th>$\mu_{RP^*}$ (%)</th>
<th>Avg. RP* in last 300 eps (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>2500</td>
<td>0.0001</td>
<td>73.83</td>
<td>92.94</td>
<td>87.30</td>
<td>91.26</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>2500</td>
<td>0.001</td>
<td>61.65</td>
<td>93.10</td>
<td>82.90</td>
<td>91.14</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>2500</td>
<td>0.0001</td>
<td>71.45</td>
<td>93.09</td>
<td>86.77</td>
<td>90.88</td>
</tr>
<tr>
<td>4</td>
<td>250</td>
<td>2500</td>
<td>0.001</td>
<td>60.30</td>
<td>91.40</td>
<td>83.69</td>
<td>88.87</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>0.001</td>
<td>0.0001</td>
<td>71.39</td>
<td>93.21</td>
<td>89.70</td>
<td>91.98</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>0.001</td>
<td>0.001</td>
<td>64.01</td>
<td>93.15</td>
<td>86.88</td>
<td>91.06</td>
</tr>
<tr>
<td>7</td>
<td>250</td>
<td>0.001</td>
<td>0.0001</td>
<td>70.95</td>
<td>93.87</td>
<td>89.58</td>
<td>92.48</td>
</tr>
<tr>
<td>8</td>
<td>250</td>
<td>0.001</td>
<td>0.001</td>
<td>62.22</td>
<td>93.57</td>
<td>87.45</td>
<td>91.50</td>
</tr>
</tbody>
</table>

Figure 4.12 shows the change in revenue performance of both agents in experiment-1. Initially, the revenue performance of both agents are low compared to their previously achieved overall average optimal revenue percentages in simulator-A, as reported in Section 4.5.2, and the DRL agent’s average optimal revenue percentage is lower among the two agents. As the testing progresses, the agents gain more experience and observations from interacting with the market. As a result, the EMSRb agent’s model estimates get more accurate and the DRL agent’s neural network weights get
Table 4.7: Results of EMSRb agent’s hyperparameter tuning experiments for adaptability test

<table>
<thead>
<tr>
<th>Exp no.</th>
<th>MUI</th>
<th>OC</th>
<th>Lowest avg. RP* (%)</th>
<th>Highest avg. RP* (%)</th>
<th>$\mu_{RP^*}$ (%)</th>
<th>Avg. RP* in last 300 eps (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>200</td>
<td>57.16</td>
<td>63.96</td>
<td>60.88</td>
<td>60.82</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>350</td>
<td>63.75</td>
<td>75.95</td>
<td>71.86</td>
<td>73.01</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>500</td>
<td>81.88</td>
<td>88.96</td>
<td>86.76</td>
<td>87.54</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>200</td>
<td>57.16</td>
<td>67.71</td>
<td>60.98</td>
<td>60.82</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>350</td>
<td>63.92</td>
<td>75.76</td>
<td>71.89</td>
<td>72.92</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>500</td>
<td>81.86</td>
<td>88.91</td>
<td>86.75</td>
<td>87.53</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>200</td>
<td>57.16</td>
<td>82.99</td>
<td>61.43</td>
<td>60.83</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>350</td>
<td>63.86</td>
<td>82.99</td>
<td>72.10</td>
<td>72.92</td>
</tr>
<tr>
<td>9</td>
<td>50</td>
<td>500</td>
<td>81.98</td>
<td>88.91</td>
<td>86.72</td>
<td>87.54</td>
</tr>
</tbody>
</table>

Figure 4.12: Optimal revenue percentage generated by agents in the event of a negative high fare product demand shock (adaptability experiment-1) in simulator-A
closer to the new optimal values, leading to an increase in their revenue performances. The average optimal revenue percentage curve of the DRL agent quickly overtakes the EMSRb agent’s one at around flight episode 250, after which it consistently remains at a higher level. In the last 300 flight episodes, this difference is 4.44% on average. The increasing trend of both curves stop at around flight episode 750. The overall average optimal revenue percentages in the entire test interval of the DRL and EMSRb agents are 89.70% and 86.76% respectively. Figure 4.13 shows the change in revenue performance of both agents in experiment-2. A similar trend can be observed in this experiment as well.

To sum up, the plots indicate that the DRL agent was able to adapt faster and better than the EMSRb agent. While engaging in exploration of the state-action space causes the DRL agent to lose some revenue in the short term, it allows faster learning of the new optimal policy in this new market environment which eventually leads to higher revenue returns in the long term and overall. A more thorough trade-off study to investigate the optimal balance of exploration and exploitation in the context of ARM can be an exciting topic for future research.
4.6 Summary

The problem of dynamic pricing in ARM is concerned with finding a policy for directly varying the prices of an airline’s fare products during the booking period based on demand and WTP of passengers to maximize the total expected flight revenues. In this study, we have formulated this problem as a MDP and solved it using DRL to find the optimal dynamic pricing policy. The fare product demands, passenger arrivals with varying wTPs, and booking cancellations were considered to be stochastic in the problem. The research methodology adopted for this study follows a real world DRL-based ARM system implementation process. The simulation results demonstrate that the DRL agent can closely match the revenue performance of exact DP methods, which indicates that it can learn the optimal dynamic pricing policy through interactions with the market despite not having any prior knowledge of the market dynamics, passenger characteristics and problem parameters. Also, the results show that the DRL agent generates higher than 90% of optimal revenues on average in all flight episodes in this uncertain market environment. Compared to the EMSRb-based ARM agent, the DRL agent produces an average optimal revenue percentage increment of 3.26%. In the event of changes in market conditions in the simulator, the DRL agent was observed to recover from the drop in revenue performance faster and subsequently produce consistently higher revenues by learning a new superior policy based on the new market conditions relative to the EMSRb agent. The DRL agent is able to achieve this better adaptability through limited and controlled exploration of the state-action space of the ARM MDP.
CHAPTER 5. CONCLUSIONS

In this doctoral research, a DRL approach was applied to ARM, a critical operations management problem faced by all airlines that determine their financial success. The primary motivation of this research was to overcome some of the limitations of traditional model-based ARM methods by using the model-free decision-making framework of DRL, investigate its capability to learn the ARM optimal policy, analyze its revenue performance, and examine its adaptability to changes in the market conditions. First, the problem of leg-based seat inventory control, and, then, the problem of dynamic pricing was tackled using DRL. The two problems are related in many aspects, but their formulations are different. For both problems, the objective was to maximize the total expected flight revenues. Multiple fare classes and products with stochastic demand, arrival of passengers with varying WTPs, and booking cancellations during the booking period have been considered in the problems. A standard air travel market simulator was developed based on the commonly adopted market dynamics, passenger behavior and problem specification for simulating passenger arrivals and booking cancellations during the booking period in the flight episodes. The simulator was used for training and testing the DRL agent. Various numerical experiments were conducted in the simulator to evaluate the performance and robustness of the proposed approach. A summary of the key contributions, key research findings of these experiments and steps for future work are presented in this chapter.
5.1 Key Contributions

To the best of our knowledge, we are the first research group in academia to explore specifically the use of DRL for ARM to overcome the shortcomings of traditional ARM methods and generate superior performance. While the airline industry have been pursuing this research direction for a few years now, their data sets, simulators and solution method details have been kept private for proprietary reasons. So, their research and results are not reproducible. We, however, are going to make our simulator, training data, test data, and complete software implementation of the DeepARM system in Python for both the seat inventory control and dynamic pricing problems publicly available on GitHub from August 2020 in an effort to motivate and accelerate further research in this direction. Another notable difference in our work is that we have considered a much higher number of decision-making instances in the problems (one decision-making instance per day), which significantly scales up the problem and makes it more realistic.

5.2 Key Research Findings

Both the seat inventory control and the dynamic pricing problems are sequential decision-making problems as they involve taking a series of actions at different points in time till the date of departure. So, they can be formulated as MDPs, which make them especially suitable for solving using DRL.

In the seat inventory control study, a total of 9 different numerical experiments were conducted by varying the mean number of passenger arrivals and cancellation rates of the fare classes to simulate different market settings. In these experiments, during training, the agent was observed to learn from its interactions with the market simulator to achieve higher percentage of optimal revenue and bring the load factor down to close to 100% despite not having any prior domain knowledge of the problem and market dynamics, such as flight capacity, fare class fares, the distribution of passenger arrivals and the cancellation rates of each fare class, etc. During testing, the DRL agent was observed to outperform the EMSRb agent in carrying out the task of seat inventory control and overbooking. In all experiments, the DRL agent achieved an average optimal revenue percentage of between 96% and 97%, whereas the EMSRb agent achieved a value of between 94.5% and 95.6%. On average,
the DRL agent achieved a 1.58% optimal revenue percentage improvement relative to the EMSRb agent. These results were made possible by the way the DRL agent controlled the seat inventory during the flight episodes. The agent was observed control seat inventory in a way that protects seats for later arriving higher fare class passengers from earlier arriving lower fare class passengers and overbook flights before departure based on the specified passenger arrival distributions and fare class cancellation rates. The state space size of the problem was in the order of approximately higher than $10^9$. Using a deep neural network to approximate the expected optimal revenues for all possible state-action combinations allowed the DRL agent to handle the large state space of the problem.

To assess the capability of the DRL agent to learn the optimal policy of the seat inventory control problem, the revenue performance of the trained DRL agent’s policy was compared with that of the optimal policy determined using the value iteration DP method. A scaled-down problem was used for this experiment to make it computationally tractable for the DP solution method. The difference in their average optimal revenue percentage and average load factor values were found to be only 0.17% and 1.35% in magnitude respectively. These results demonstrate that the DRL agent, given proper training, can learn the optimal policy of the ARM MDP.

For the dynamic pricing study, the research methodology followed closely resembles a real world DRL-based ARM system implementation process. Two market simulators were used, one acted as the actual market, and the other as the training simulator for the DRL agent. The state space size of the problem was in the order of approximately higher than $10^{10}$. Several different types of numerical experiments were performed. Firstly, hyperparameter tuning experiments were carried out for both the DRL and EMSRb agents with different hyperparameter settings to fine tune their hyperparameter values. As for the previous study, among all the different neural network architectures considered, a deep neural network with three hidden layers and 256 nodes per hidden layer was found to generate the highest average optimal revenue percentage. Also, the simple DQL learning algorithm and its more advanced variants were found to produce similar performances. Secondly, the agents were tested in the actual market simulator. The DRL agent was observed to generate a average optimal revenue percentage gain of 3.26% on average relative to the EMSRb agent. Thirdly, the DRL agent’s
performance was compared to that of the optimal policy derived using the value iteration DP method for a scaled-down version of the problem. The optimality gap was found to be only 0.12%, which indicates the DRL agent, given proper training, can learn the optimal policy of the ARM dynamic pricing MDP, without requiring as inputs the explicit transition function and reward function of the problem unlike its DP counterparts. Lastly, market perturbations in the form of a demand disruption and change in cancellation behavior of passengers were introduced in the actual market simulator to test the adaptability of both agents. As the agents were not given any initial knowledge of the change in market conditions, they initially followed their previous policy, leading to a degradation of their revenue performance. However, after an initial interval of about 50 episodes, the revenue performance of both agents started climbing again. This rate of recovery or adaptability strongly depended on their learning abilities. For the EMSRb agent, the learning occurred indirectly through update of its model estimates. For the DRL agent, the learning occurred directly based on the revenue signals and observations it received from its interactions with the market. By following a controlled exploration policy, the DRL agent was observed to recover faster than the EMSRb agent, surpassing its revenue performance within the first around 250 flight episodes, and learn a superior policy based on this new market setting, leading the EMSRb agent by 4.44% in the last 300 flight episodes and 2.94% in all episodes on average in terms of the optimal revenue percentage generated.

In conclusion, the DRL agent was observed to generate higher revenues than the EMSRb method in all experiments. In the smaller-scale versions of the problems, the DRL agent was observed to match the revenue performance of exact DP methods. This research has also demonstrated that a DRL-based ARM system have the following four major advantages over traditional ARM approaches:

1. **Integrity**: Unlike the traditional approaches, the DRL agent does not require any external data or models, such as demand, and passenger arrival, choice and cancellation models, after it has been trained in the simulator. Hence, once it is deployed in the real world market, its performance does not depend on the accuracy of such external data or models.

2. **Scalability**: As the scale of the problem increases, the number of decision variables and size
of the state-action space increases drastically, rendering traditional methods, like exact DP methods and dynamic optimization approaches, impractical. A DRL approach can however handle large state-action spaces of the problem by using a deep neural network to approximate the expected optimal revenue for any state-action pair. The deep neural network also allows the agent to generalize from limited experience, meaning that it is not necessary for the DRL agent to experience all state-action pairs to get an accurate approximation.

3. **Optimality**: Given sufficient interactions with the market and exploration of the state-action space, the DRL agent can directly learn the true optimal ARM policy, whereas the optimality of the policy derived using traditional methods depends on the accuracy of the model estimates, which are never known with complete accuracy, and the policy derived using heuristics-based traditional methods are not optimal.

4. **Adaptability**: Because the agent is continuously learning from its interactions, it can autonomously adapt to changes to the market. Through sufficient exploration, it can quickly learn a new optimal policy when market conditions change.

### 5.3 Future Research Directions

The promising results found in this research is expected to serve as strong motivation for future research that builds on this current work. Some of the potential areas of research that form exciting topics for future research are listed below.

1. The focus of this research was on leg-based ARM, where individual flight leg revenues in the airline network are optimized separately. So, one natural extension to this current work would be to apply DRL to network ARM to maximize the total revenue generated in the entire network. For small networks with a limited number of flight legs, a single DRL agent may still be adequate for practicing ARM. The state representation would have to be extended to include the booking counts of the various fare classes in all flight legs. The action space would also have to be extended to allow the agent control of seat inventory and pricing in all flight
legs. For large networks with many connecting flight legs, one promising DRL approach for this problem would be multi-agent reinforcement learning, where multiple cooperative DRL agents may be employed to practice ARM, with each assigned to one or more flight legs, and coordinate their policies with each other.

2. Another extension of this work involves adding one or more competitors in the market simulator and applying game theoretic principles to investigate if the DRL agent can determine the Nash equilibrium optimal ARM strategy. To make the agent more aware of its competitors in the market, the state representation may be extended by including additional state variables denoting competitors’ offerings and prices at each time step.

3. During adaptability testing, the DRL agent engaged in controlled exploration of the state-action space, causing it to lose some revenue in the short term, but allowing it to learn faster the new optimal policy in the new market environment which eventually led to higher revenue returns in the long term and overall. As air travel market characteristics, such as demand, competition and passenger behavior, typically change with time, a more thorough trade-off study to investigate the optimal balance of exploration and exploitation in the context of ARM is a worthwhile future research endeavor.

4. As the adaptability of the DRL agent in the true market simulator highly depends on the efficiency of its learning, another beneficial future research direction is to optimize the learning of the agent by exploring the use of more efficient learning algorithms or customized algorithms to have better generalization and/or parallel computing on simulators to speed up learning.

5. As the airline industry is experiencing changes in business practices and industry standards in recent times [Belobaba et al. (2017)], another promising future research pursuit involves incorporating some of these changes in the DRL-based ARM framework and simulator to further improve its practical applicability and forward compatibility. It would be especially interesting to include dynamic adjustment of price points, continuous pricing and dynamic offer generation capabilities in the system. For dynamic adjustment of price points, the agent’s
action space would need to be modified to allow it mark up and down the initial set of price points. For continuous pricing, the agent’s action space would need to be made continuous to allow it to select any arbitrary price point. For dynamic offer generation, the agent’s action space would need to be extended to include actions that allow it to dynamically create fare products by bundling itineraries and add-on services such as extra legroom seats, checked bags, and lounge access.

6. Another exciting future research direction is to adopt the DRL-based ARM system for use in other industries that also control their inventory and price their products in a way similar to the airline industry, such as the hospitality, rental car and advertising industries.
BIBLIOGRAPHY


