# A Deep Reinforcement Learning Approach to Seat Inventory Control for Airline Revenue Management

**Syed A.M. Shihab***
*Kent State University, Kent, OH 44242, USA*

**Peng Wei**†
*George Washington University, Washington, DC 20052, USA*

August 2020

**Abstract**

Commercial airlines use revenue management systems to maximize their revenue by making real-time decisions on the booking limits of different fare 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 external 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 policy of the seat inventory control problem. 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 space of the problem. Multiple fare classes with stochastic demand, passenger arrivals and booking cancellations have been considered in the problem. 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 revenue generated by the agent in different simulated market scenarios was found to be close to the maximum possible flight revenues and surpass that produced by the expected marginal seat revenue-b (EMSRb) method.

**Keywords** — Airline revenue management · Seat inventory control · Deep reinforcement learning

## 1 Introduction

Few markets are as fiercely competitive as the current air travel market. This heightened competition dates back to the deregulation of the airline industry in 1978, which allowed US airlines to freely set up their route network and quote fares for their itineraries. Since then, airlines have been relying on *airline revenue management* (ARM) systems, a decision support tool designed for maximizing the total expected revenues generated from all their flights. Considering the large scale nature of operations of traditional network airlines, even small revenue increments per flight provided by ARM systems is significant: the sale of only one seat per flight at full price instead of the discount rate can contribute an additional $50

*Assistant Professor, College of Aeronautics and Engineering, sshihab@kent.edu
†Assistant Professor, Department of Mechanical & Aerospace Engineering, pwei@gwu.edu

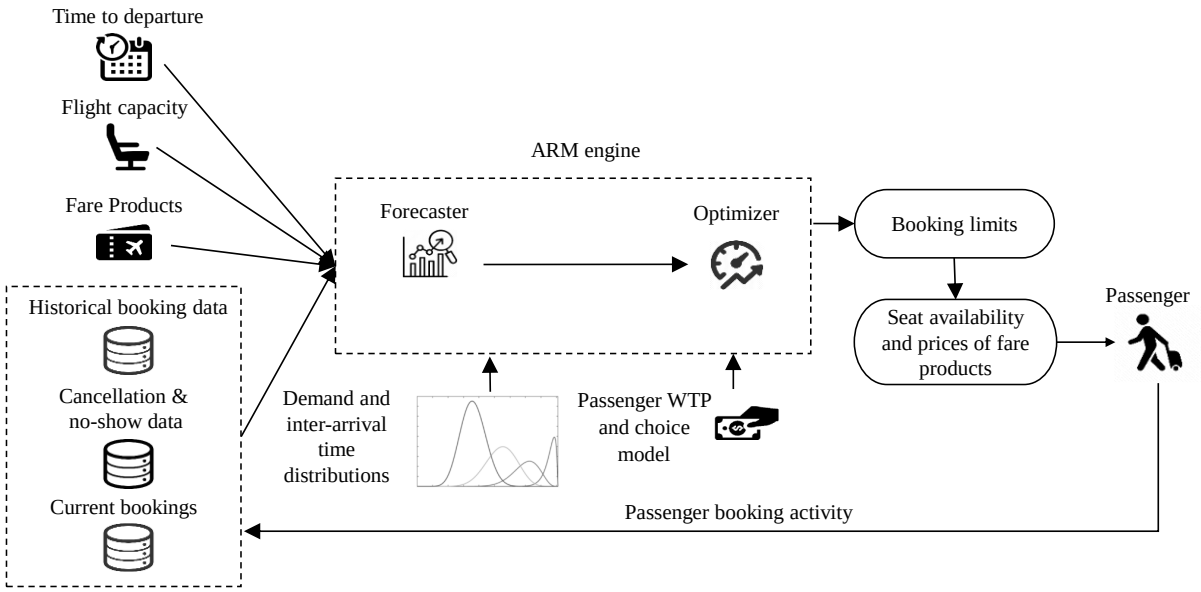


Figure 1: Traditional model-based ARM system architecture

million to the airline's annual revenues [Cross (2011)]. The next few subsections discuss the construction and operation of traditional model-based ARM systems, the motivation for this research and the outline of the rest of the paper.

## 1.1 Traditional Model-Based ARM System

The architecture of a traditional model-based ARM system is depicted in Figure 1. These 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. These 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 heuristic rule-based decision models. When a passenger arrives to make a booking, they see the fares of the open fare classes and their seat availability and decide 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.

## 1.2 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 polices 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. The ARM problem of seat inventory control has been examined in this paper. To investigate the potential advantages of using the DRL method for this problem, a DRL-based ARM system was developed to learn the optimal seat inventory control policy. Additionally, an air travel market simulator was built to train and test the performance of the DRL-based ARM system relative to traditional ARM methods.

## 1.3 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)].

### 1.3.1 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 $s_t$ to $s_{t+1}$ based on $T(s_t, a, s_{t+1})$ and receive a reward $r_t = R(s_t, a, s_{t+1})$. 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)]:

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

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 it is given by the Bellman equation:

$$U^*(s_t) = \max_{a} \sum_{s_{t+1} \in S} T(s_t, a, s_{t+1})(R(s_t, a, s_{t+1}) + \gamma U^*(s_{t+1})). \tag{2}$$

The equation above is called the *value function* for the optimal policy. In the value function, the expected immediate (current time step) reward at state $s_t$ is given by $\sum_{s_{t+1} \in S} T(s_t, a, s_{t+1}) R(s_t, a, s_{t+1})$ and the expected discounted future reward at state $s_t$ by $\sum_{s_{t+1} \in S} T(s_t, a, s_{t+1}) \gamma U^*(s_{t+1})$. 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:

$$Q^*(s_t, a) = \sum_{s_{t+1} \in S} T(s_t, a, s_{t+1})(R(s_t, a, s_{t+1}) + \gamma \max_{a'} Q^*(s_{t+1}, a'))). \tag{3}$$

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:

$$\pi^*(s_t) = \arg\max_{a} \sum_{s_{t+1} \in S} T(s_t, a, s_{t+1})(R(s_t, a, s_{t+1}) + \gamma U^*(s_{t+1})). \tag{4}$$

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

$$\pi^*(s_t) = \arg\max_{a} Q^*(s_t, a), \tag{5}$$

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

### 1.3.2 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

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a)), \tag{6}$$

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.

### 1.3.3 Balancing Exploration and Exploitation

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). Two of the commonly used exploration policies are the $\epsilon$-greedy and linear annealed $\epsilon$-greedy policies. An $\epsilon$-greedy policy specifies the agent to choose a random action with probability $\epsilon$ and the exploitative action with probability $1-\epsilon$. In a linear annealed $\epsilon$-greedy policy, the value of $\epsilon$ is linearly decreased with time (experience) so that the agent performs high exploration at the beginning and high exploitation at the end.

### 1.3.4 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.

In deep Q-learning (DQL), this generalization is achieved by using a deep neural network to approximate the Q-values [Mnih et al. (2013); Mnih et al. (2015)]. As DQL involves the use of a deep neural network, it is considered to be a type of DRL algorithm. The inputs to the deep neural network are the state variables and the outputs are the Q-values. The deep neural network can approximate highly nonlinear Q-functions as it is typically composed of several hidden layers. Nonlinearity is introduced in the function approximation of neural networks using activation functions, such as the sigmoid, hyperbolic tangent (Tanh) and rectified linear unit (ReLU) 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)]. In DQL, Q-learning with backpropagation and mini-batch gradient descent is used to update the weights of the neural network such that the loss function is minimized. After convergence is reached during training, the optimal policy can be found using Eq. 5.

## 1.4 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 fare class bookings and cancellations and progression of time.

## 1.5 Outline of Paper

The remainder of the paper is organized as follows. A literature review of related work is given in Section 2. Section 3 formally defines the ARM seat inventory control problem considered in this paper. The

subsequent sections describe the air travel market simulator, the MDP formulation of the problem, the solution method, and the results successively. Finally, the key research findings of this work and future research directions are summarized in the concluding section.

## 2 Related Work

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 traditional 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 past ARM research 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 a lack of competition in the O-D markets [Talluri and Van Ryzin (2006)]. Until recently, the vast majority of ARM research has focused more on refining previously developed ARM methods and relaxing some of their assumptions to make them more realistic rather than developing entirely new methods.

Over the last two decades, there have been a very limited number of attempts at using reinforcement learning for ARM. Gosavi et al. (2002) were among the first to pursue this research direction by applying their novel RL algorithm called $\lambda$-SMART to the single leg ARM seat inventory control problem. Along similar lines, Lawhead and Gosavi (2019) developed a bounded actor–critic RL algorithm and applied it to the same problem. In both papers, the RL agent was shown to outperform the EMSRb ARM method. Some of the key differences between their work and ours are:

a) They derived the $\lambda$-SMART algorithm, a multi-step temporal difference-based algorithm, and used it as their solution method whereas we used deep Q-learning, a popular DRL algorithm, as the solution method. The use of the deep neural network in our solution method allowed us to handle more complexity and stochasticity in the ARM problem.

b) They treated time as continuous and formulated the ARM problem as a semi-MDP, assuming that airlines make seat inventory control decisions upon the arrival of each passenger, whereas we formulated the ARM problem as a discrete time MDP and considered the decision-making to occur at the beginning of each discrete time step or data collection point (a decision making instance) within the booking period, which more closely resembles a real-life ARM process.

c) They used a regular Poisson process to simulate the arrival of fare class passengers, while we used the NHPP, a process that has been been shown to fit well real-world airline passenger arrival data [Beckmann and Bobkoski (1958); Weatherford et al. (1993); and Bertsimas and De Boer (2005)]. Also, they used a constant cancellation probability to model the passenger booking cancellations. In real-life, the passenger cancellation probability varies with time along the booking period, being significantly high near flight departure and low at other times. This passenger cancellation behavior was taken into account in our problem.

d) Upon any passenger arrival, they assumed that the passenger makes a booking request of their desired fare class and allowed the RL agent to subsequently take one of two actions, accept the booking request or deny it. The RL agent was allowed to deny a booking request even if there were seats available on the flight. In real-life, however, passengers typically arrive at a booking platform

(such as an airline's online website), checks for seat availability and fares, and makes a decision to either book a seat or leave the booking platform; if a passenger decides to book a seat of a certain fare class while there are seats available (the fare class is open), the passenger's booking request typically do not get denied in real-life. This passenger booking process was taken into account in our problem by allowing the DRL agent to open and close fare classes at the beginning of each time step rather than accept or deny individual booking requests.

More recently, Bondoux et al. (2020) tackled the ARM dynamic pricing problem using DRL. The problem involved a fenceless fare structure and assumed there were no cancellations and overbooking. In their work, the DRL agent was shown to be able to learn the optimal ARM dynamic pricing policy and outperform a traditional ARM system in a duopoly competition. In this paper, we address the ARM seat inventory control problem with multiple fare classes, considering both passenger cancellations and overbooking, which leads to a significantly different formulation of the ARM MDP.

# 3 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 (or revenue value buckets) with stochastic demands, a booking period of 182 days, and a flight capacity ($\kappa$) of 100 have been considered in the problem. Each virtual fare class is assumed to be a cluster of several O-D fare classes using the same flight leg and having similar network revenue values, and formed based on the method pioneered by Smith and Penn (1988). The three virtual fare classes will simply be referred to as the high (H), middle (M) and low (L) fare class henceforth. 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 in anticipation of future cancellations. The fare class booking arrival and cancellation distribution parameters are described in Sections 4.1 and 4.2 below. The booking period 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.

# 4 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 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)].

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.

## 4.1 Passenger Arrival Model

The passenger arrival of each fare class has been modeled as a NHPP. The time-varying mean arrival rate of the different fare classes are determined using

$$\lambda_i(t) = \int_t^{t+1} \lambda_i'(\tau; \widehat{\alpha}_i, \widehat{\beta}_i)\ \widehat{A}_i\ d\tau, \tag{7}$$

where $\lambda_i(t)$ is the mean arrival rate of fare class $i$ at time step $t$, $\lambda_i'(\tau; \widehat{\alpha}_i, \widehat{\beta}_i)$ is the probability density function value of a beta distribution with shape parameters $\widehat{\alpha}_i$ and $\widehat{\beta}_i$ corresponding to fare class $i$ at time $\tau$, and $\widehat{A}_i$ 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

$$\lambda_i'(\tau) = \frac{(\frac{\tau}{182})^{\widehat{\alpha}_i - 1}(1 - \frac{\tau}{182})^{\widehat{\beta}_i - 1}}{\mathrm{B}(\widehat{\alpha}_i, \widehat{\beta}_i)}, \tag{8}$$

where

$$\mathrm{B}(\widehat{\alpha}_i, \widehat{\beta}_i) = \frac{\Gamma(\widehat{\alpha}_i)\Gamma(\widehat{\beta}_i)}{\Gamma(\widehat{\alpha}_i + \widehat{\beta}_i)} \tag{9}$$

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 1. The total number of expected passengers is between 150 and 156, a factor of about 1.5 higher than the capacity, 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, $\widehat{A}_i$ is replaced by the total number of passenger arrivals, which is typically modeled with a gamma distribution. In our model, we have kept $\widehat{A}_i$ deterministic. The shape parameters $\widehat{\alpha}_i$ and $\widehat{\beta}_i$ of the beta distributions determine the passenger arrival patterns of the fare classes. Table 2 lists the shape parameter values used in the simulator.

Figure 2 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

Table 1: The fare class mean arrivals per episode in different mean arrival settings

| MAS | $\widehat{A}_H$ | $\widehat{A}_M$ | $\widehat{A}_L$ |
|---|---|---|---|
| 1 | 16 | 40 | 100 |
| 2 | 25 | 40 | 90 |
| 3 | 30 | 40 | 80 |

Table 2: The beta distribution shape parameter values of the fare classes

| Fare class ($i$) | $\widehat{\alpha}_i$ | $\widehat{\beta}_i$ |
|---|---|---|
| H | 28.35 | 1.65 |
| M | 15.4 | 4.6 |
| L | 11.55 | 18.45 |

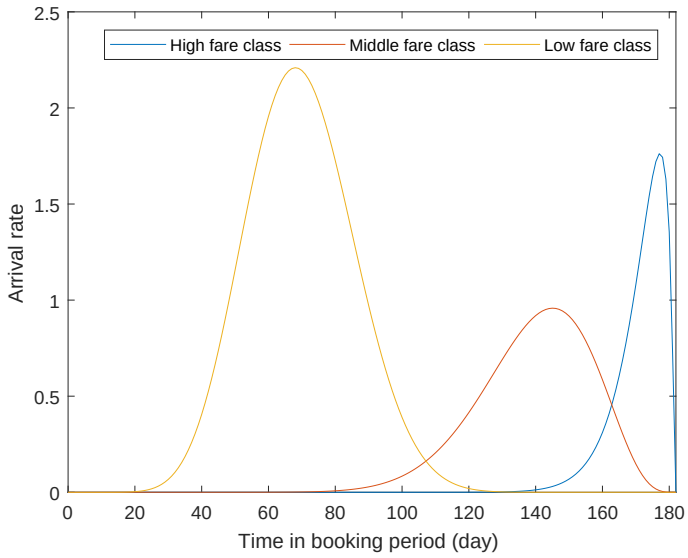


Figure 2: Plot of passenger arrivals rates of different fare classes for the second MAS

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.

## 4.2 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 [Iliescu et al. (2008); Petraru (2016)]. 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 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.

Table 3: The fare class cancellation rates in different cancellation probability settings

| CPS | H | M | L |
|---|---|---|---|
| 1 | 5% | 1% | 0% |
| 2 | 10% | 5% | 0% |
| 3 | 15% | 10% | 0% |

### 4.3 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

$$r_e^* = \sum_{i \in I} n_{i,e}^* f_i, \tag{10}$$

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 average 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

$$n_{i,e}^* = \min\left(A_{i,e}^T - C_{i,e}^T, \widehat{\kappa}_{i,e}\right), \tag{11}$$

$$\widehat{\kappa}_{i,e} = \kappa - \sum_{i \in \{H,\ldots,i-1\}} n_{i-1,e}^*, \tag{12}$$

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 $\widehat{\kappa}_{i,e}$ 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, $\widehat{\kappa}_{H,e} = \kappa$ 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.

## 5 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.

### 5.1 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

$$s = (b_H, b_M, b_L, t), \tag{13}$$

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 infinite size as the Poisson arrival distributions are partially unbounded. Note that the states can also be represented as $s_t$, as was done in Section 1.3.1.

## 5.2 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

$$a \in \{\widehat{a}_1, \widehat{a}_2, \widehat{a}_3\}. \tag{14}$$

The interpretation of the different actions is given in Table 4. Action $\widehat{a}_1$ specifies that only the high fare class is open and the other two lower classes are closed. Action $\widehat{a}_2$ specifies that both the high and middle fare classes are open and the lowest fare class is closed. Finally, action $\widehat{a}_3$ 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 4: Actions the agent can take to control the seat inventory

| Action | $H$ | $M$ | $L$ |
|---|---|---|---|
| $\widehat{a}_1$ | open | close | close |
| $\widehat{a}_2$ | open | open | close |
| $\widehat{a}_3$ | open | open | open |

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 $\widehat{a}_1$ is useful for saving the remaining seats for the high fare class passengers in the current time step. Similarly, action $\widehat{a}_2$ allows the agent to save the remaining seats for the high and middle fare class passengers in the current time step. The third action $\widehat{a}_3$ 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.

## 5.3 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

$$\Delta b_{t,i} = \beta_{t,i} - C_{t,i}, \tag{15}$$

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,

$$\beta_{t,i} = \begin{cases} \alpha_{t,i}, & \text{if fare class } i \text{ is open at } t \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Once the terminal state ($t = 182$) is reached, all actions will lead to the ending of the episode.

### 5.4 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:

$$R(s, s') = \begin{cases} \sum\limits_{i \in I} \Delta b_{t,i} f_i, & \text{for } 0 \leq t \leq 181 \\ - \sum\limits_{i \in \{L,M\}} \eta_i B f_i - 675\eta_H, & \text{if } t = 182, \end{cases} \tag{17}$$

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 $- \sum\limits_{i \in \{L,M\}} \eta_i B f_i - 675\eta_H$, 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

$$\eta^T = \max\left(\sum_{i \in I} b_i - \kappa, 0\right), \quad t = 182. \tag{18}$$

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. 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:

$$\eta_L = \begin{cases} \eta^T, & \text{if } \eta^T \leq b_L \\ b_L, & \text{if } \eta^T > b_L \end{cases} \tag{19}$$

$$\eta_M = \begin{cases} \eta^T - \eta_L, & \text{if } \eta^T - \eta_L \leq b_M \\ b_M, & \text{if } \eta^T - \eta_L > b_M \end{cases} \tag{20}$$

$$\eta_H = \eta^T - \eta_L - \eta_M \tag{21}$$

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.

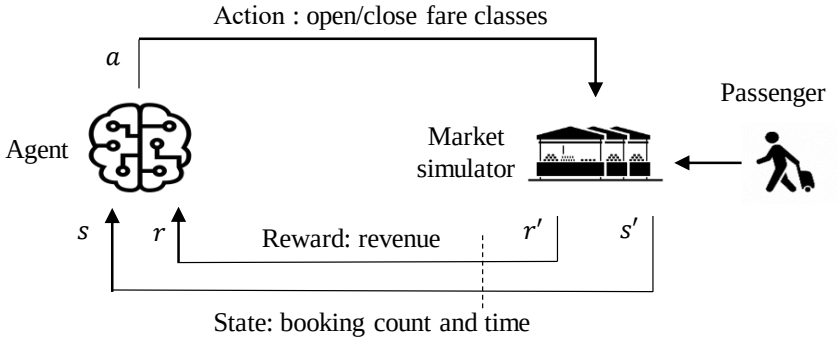


Figure 3: The agent-market simulator interaction

# 6 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 Section 1.3. The interactions taking place between the agent and the market simulator is shown in Fig. 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 time step.

## 6.1 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.

## 6.2 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 $\varepsilon$-greedy policy was chosen as the exploration policy and the value of $\varepsilon$ 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 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.

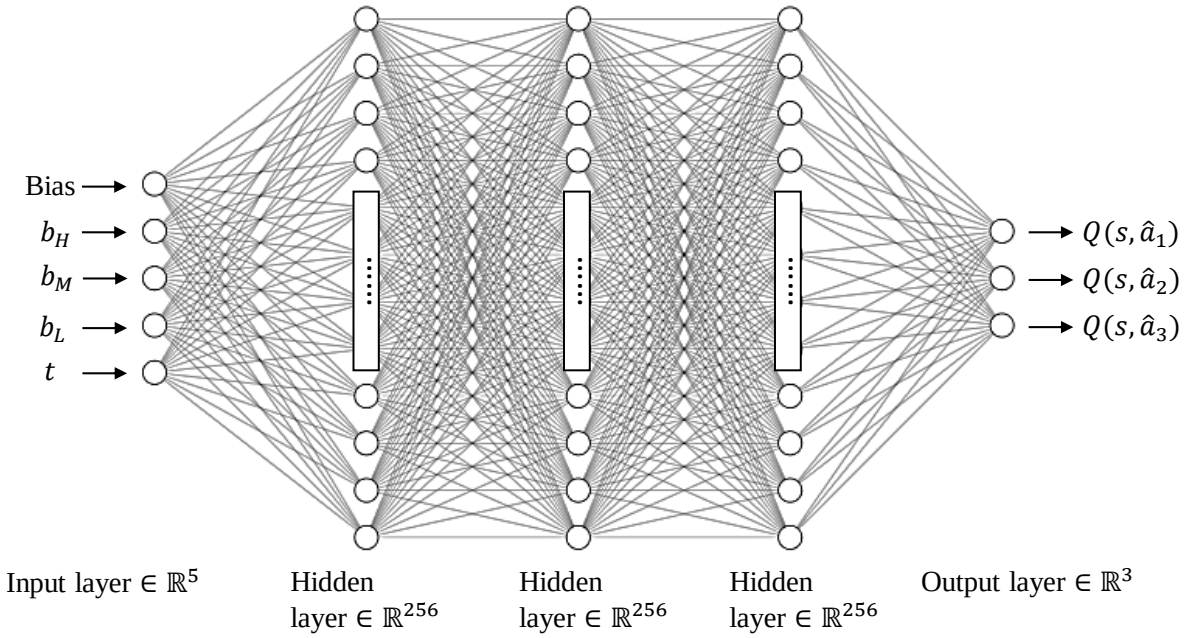


Figure 4: The configuration of the deep neural network used to approximate Q-values

# 7 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 was 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 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.

## 7.1 Training Results

The training plots for the numerical experiment with MAS=1 and CPS=1 are presented in Figs. 5 and 6. 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.

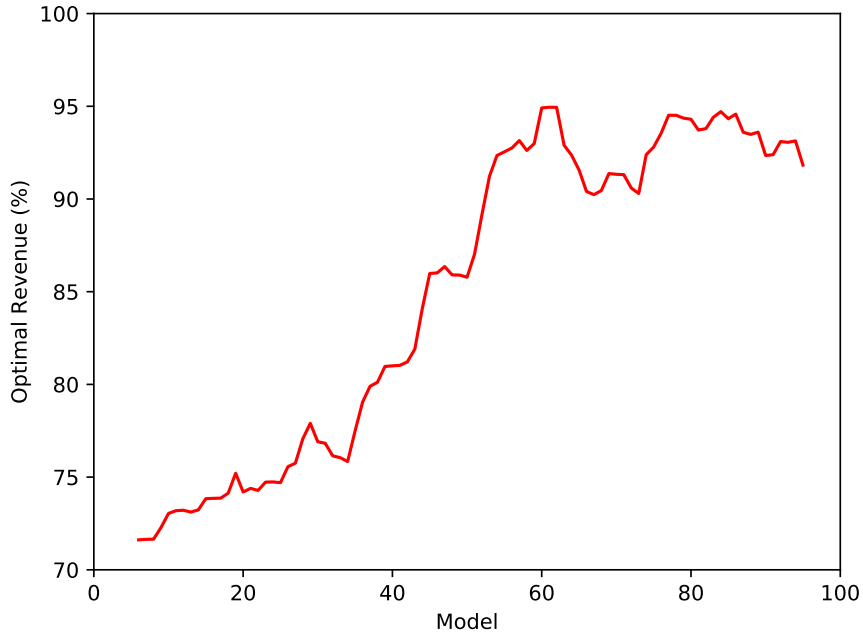


Figure 5: Average optimal revenue percentage generated by agent during training for MAS = 1 and CPS = 1

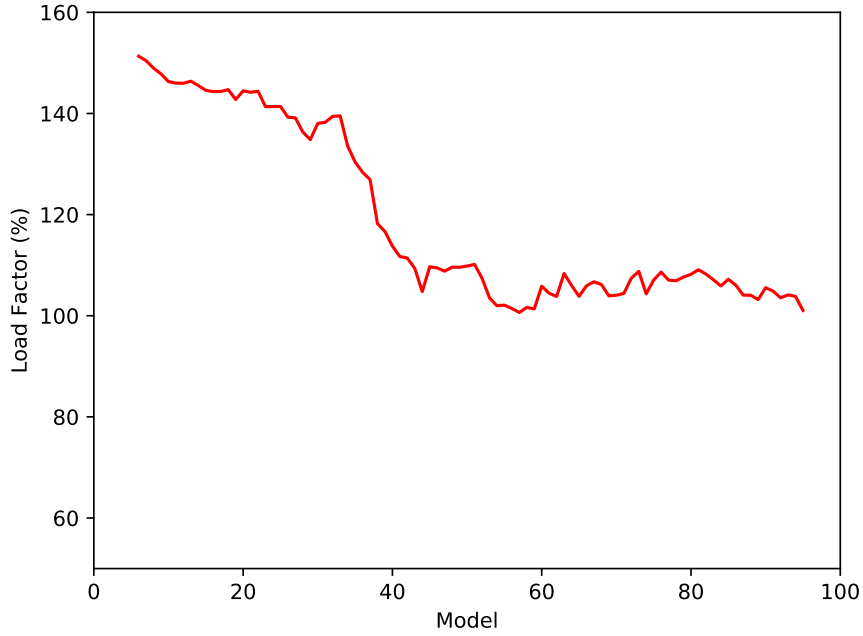


Figure 6: Load factor achieved by agent during training for MAS = 1 and CPS = 1

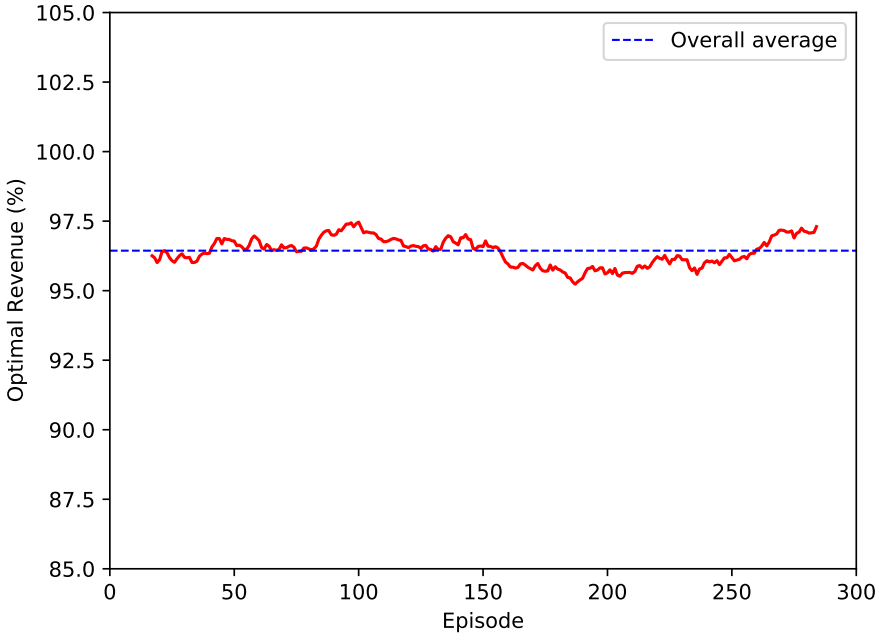


Figure 7: Optimal revenue percentage generated by agent during testing for MAS = 1 and CPS = 1

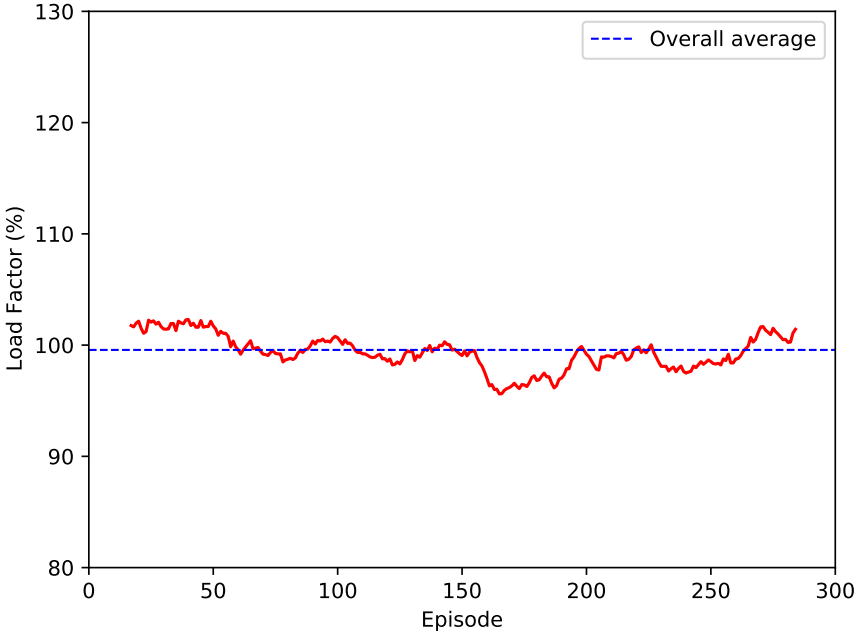


Figure 8: Load factor achieved by agent during testing for MAS = 1 and CPS = 1

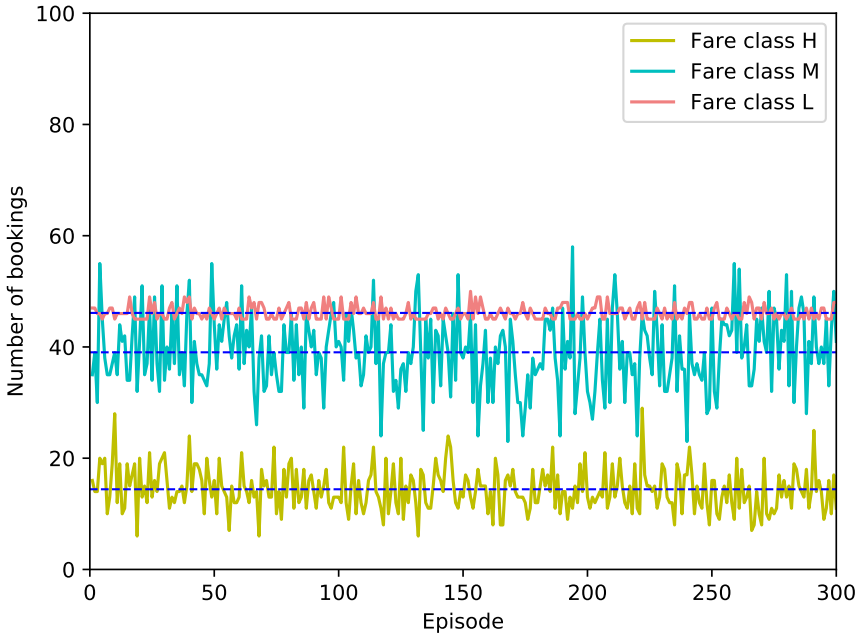


Figure 9: Fare class booking plot of agent during testing for MAS = 1 and CPS = 1

## 7.2 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. 7, 8 and 9. 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 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. 9 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.

## 7.3 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 policy) is also used to benchmark the performance of the DRL agent, as it is considered to be a robust leg-based ARM method with strong revenue performance [Belobaba (2016)]. 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 5 and 6 respectively. The improvement in performance of the DRL agent relative to the EMSRb agent is listed in Table 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 ($\sigma_R$) and the 95% confidence interval for the revenue generated (95% $\text{CI}_R$).

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.

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 lower booking limits for the low and middle fare class passengers to protect seats for the later arriving high fare class passengers, resulting in a lower $\mu_{LF}$. 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%.

Table 5: Performance of the DRL agent in the different numerical experiments

| CPS | MAS | $\mu_{RP^*}$ (%) | $\sigma_{RP^*}$ (%) | $\mu_{LF}$ (%) | $\sigma_{LF}$ (%) | $\mu_R$ (\$) | $\sigma_R$ (\$) | 95% $\text{CI}_R$ (\$) |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 96.44 | 2.68 | 99.57 | 8.01 | 17584.33 | 1445.43 | ±164.50 |
| 1 | 2 | 96.57 | 2.51 | 98.91 | 8.32 | 19961.33 | 1763.64 | ±200.72 |
| 1 | 3 | 96.96 | 2.24 | 102.52 | 7.96 | 20900 | 1451.57 | ±165.20 |
| 2 | 1 | 96.35 | 2.5 | 104.73 | 6.75 | 16928.33 | 947.79 | ±107.87 |
| 2 | 2 | 96.97 | 2.12 | 102.36 | 7.09 | 19191 | 1361.62 | ±154.96 |
| 2 | 3 | 96.65 | 2.61 | 98.38 | 8.3 | 20007.33 | 1620.91 | ±184.47 |
| 3 | 1 | 96.73 | 2.47 | 100.87 | 6.96 | 16102.33 | 1113.63 | ±126.74 |
| 3 | 2 | 96.76 | 2.59 | 97.6 | 6.94 | 18134.33 | 1587.09 | ±180.62 |
| 3 | 3 | 96.71 | 2.42 | 104.01 | 7.22 | 18865 | 1152.94 | ±131.21 |

Table 6: Performance of the EMSRb agent in the different numerical experiments

| CPS | MAS | $\mu_{RP^*}$ (%) | $\sigma_{RP^*}$ (%) | $\mu_{LF}$ (%) | $\sigma_{LF}$ (%) | $\mu_R$ ($) | $\sigma_R$ ($) | 95% $CI_R$ ($) |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 94.63 | 3.64 | 96.42 | 4.28 | 17228.67 | 1165.18 | ±132.61 |
| 1 | 2 | 94.9 | 3.21 | 96.09 | 4.82 | 19590.67 | 1491.68 | ±169.77 |
| 1 | 3 | 95.53 | 3.12 | 96.29 | 4.72 | 20585 | 1436.11 | ±163.44 |
| 2 | 1 | 95.1 | 3.23 | 97.12 | 3.94 | 16711.33 | 1039.23 | ±118.27 |
| 2 | 2 | 95.18 | 3.16 | 97.07 | 4.54 | 18832.33 | 1338.98 | ±152.39 |
| 2 | 3 | 95.06 | 3.36 | 96.7 | 4.86 | 19653.33 | 1354.56 | ±154.16 |
| 3 | 1 | 94.98 | 3.23 | 97 | 4.3 | 15804 | 1064.35 | ±121.13 |
| 3 | 2 | 95.18 | 3.06 | 97.23 | 4.61 | 17811.33 | 1259.84 | ±143.38 |
| 3 | 3 | 95.34 | 3.03 | 97.33 | 4.62 | 18600.33 | 1205.75 | ±137.22 |

Table 7: Performance of the DRL agent relative to the EMSRb agent in the different numerical experiments

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

## 7.4 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. 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 infinite (as discussed in Section 5.1), which would prevent the exact DP methods from reaching convergence. 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 Poisson arrival distributions are truncated at the points beyond which the arrival probabilities are less than the order of $10^{-2}$. This reduces the size of the state space size to the order of $10^5$. 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 average revenue values 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

$$T(s,a,s') = \begin{cases} P(\Delta b_{t,H})P(\Delta b_{t,L}), & \text{if } t'-t=1 \\ 1, & \text{if } t'=10,\ t=0,\ b_H=0 \text{ and } b_L=0\ (s=\text{initial state}) \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

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:

$$P(\Delta b_{t,H}) = \frac{\lambda_H(t)^{\Delta b_{t,H}} e^{-\lambda_H(t)}}{\Delta b_{t,H}!} \tag{23}$$

$$P(\Delta b_{t,L}) = \begin{cases} \frac{\lambda_L(t)^{\Delta b_{t,L}} e^{-\lambda_L(t)}}{\Delta b_{t,L}!}, & \text{if } a=1 \\ 1, & \text{if } a=0,\ \Delta b_{t,L}=0 \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

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:

$$R(s,s') = \begin{cases} \sum_{i \in I} \Delta b_{t,i} f_i, & \text{for } 0 \leq t \leq 9 \\ -\eta_L B f_L - \eta_H B f_H, & \text{if } t = 10. \end{cases} \tag{25}$$

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. 10 and 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 $\mu_{LF}$ achieved by the DRL agent during testing were 92.86% and 100.99% 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 7.3. 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.

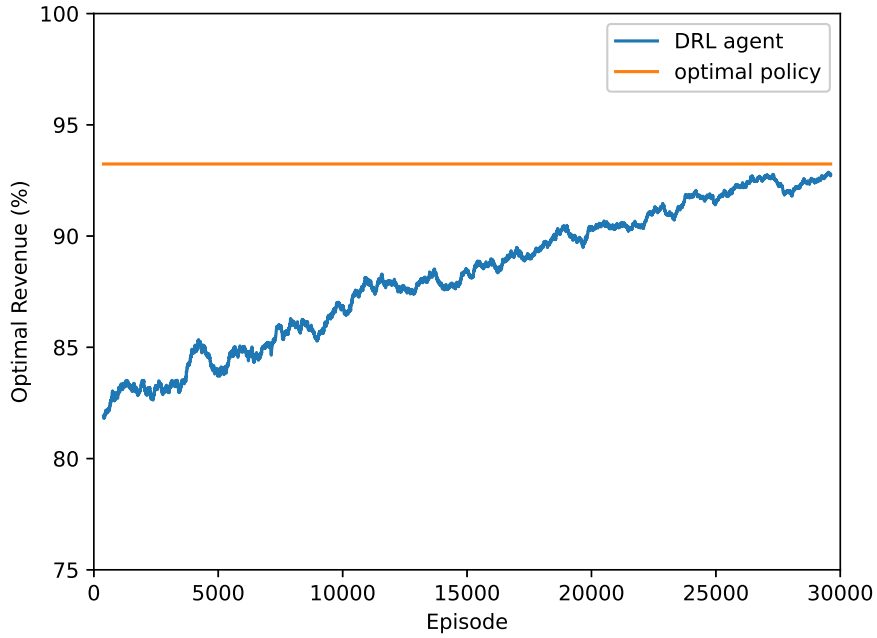


Figure 10: Optimal revenue percentage generated by agent during training for the ARM MDP with 2 fare classes

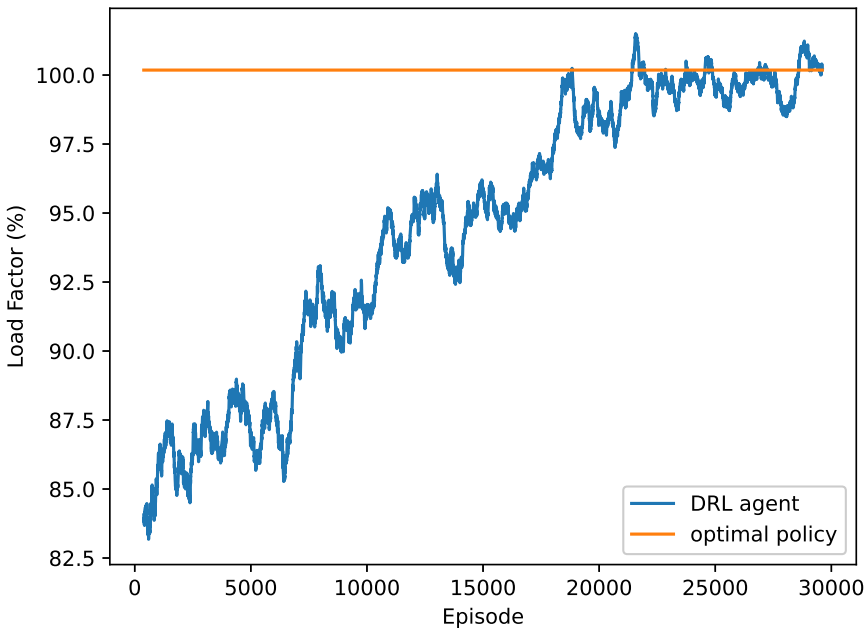


Figure 11: Load factor achieved by agent during training for the ARM MDP with 2 fare classes

# 8 Implementation of a DRL-based Airline Revenue Management System in the Real World

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. 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.

# 9 Conclusion

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 control policy for airline revenue management with overbooking. Multiple fare classes with stochastic fare class demand, passengers arrivals and cancellations were considered in the problem. 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. Although the state space size was infinite, the Use of a deep neural network allowed the DRL agent to approximate the expected optimal revenues for all possible state-action combinations. The results show that the DRL agent generates close to optimal revenues in all flight episodes in all numerical experiments, outperforming the 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%.

A DRL-based ARM system have a number of inherent advantages over traditional ARM approaches. Firstly, unlike the traditional ARM 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. Secondly, the use of a deep neural network allows it to handle large and continuous state spaces of the ARM problem. Thirdly, given enough interactions with the market, the agent can directly learn the true optimal ARM policy. Lastly, because the agent is continuously learning from its interactions, it can autonomously adapt to changes to the market.

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 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. 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.

# References


Robert G Cross. *Revenue management: Hard-core tactics for market domination*. New York, NY: Currency, 2011.

Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. Cambridge, MA: MIT press, 2015.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Luo Li and Ji-Hua Peng. Dynamic pricing model for airline revenue management under competition. *Systems Engineering-Theory & Practice*, 27(11):15–25, 2007.

Kalyan T Talluri and Garrett J Van Ryzin. *The theory and practice of revenue management*, volume 68. Springer Science & Business Media, 2006.

Lawrence R Weatherford, Samuel E Bodily, and Phillip E Pfeifer. Modeling the customer arrival process and comparing decision rules in perishable asset revenue management situations. *Transportation Science*, 27(3):239–251, 1993.

Guillermo Gallego and Garrett Van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science*, 40(8):999–1020, 1994.

Abhijit Gosavi, Naveen Bandla, and Tapas K Das. A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking. *Institute of Industrial Engineers (IIE) transactions*, 34(9):729–742, 2002.

Ryan J Lawhead and Abhijit Gosavi. A bounded actor–critic reinforcement learning algorithm applied to airline revenue management. *Engineering Applications of Artificial Intelligence*, 82:252–262, 2019.

Martin J Beckmann and F Bobkoski. Airline demand: An analysis of some frequency distributions. *Naval research logistics quarterly*, 5(1):43–51, 1958.

Dimitris Bertsimas and Sanne De Boer. Simulation-based booking limits for airline revenue management. *Operations Research*, 53(1):90–106, 2005.

Nicolas Bondoux, Anh Quan Nguyen, Thomas Fiig, and Rodrigo Acuna-Agost. Reinforcement learning applied to airline revenue management. *Journal of Revenue and Pricing Management*, pages 1–17, 2020.

Barry C Smith and CW Penn. Analysis of alternate origin-destination control strategies. In *AGIFORS PROCEEDINGS*, Cape Cod, MA, 1988.

PODS Research, LLC. The Passenger Origin-Destination Simulator (PODS). http://podsresearch.com/pods.html, 2017. Accessed: 2020-06-29.

Dan C Iliescu, Laurie A Garrow, and Roger A Parker. A hazard model of us airline passengers' refund and exchange behavior. *Transportation Research Part B: Methodological*, 42(3):229–242, 2008.

Oren Petraru. *Airline passenger cancellations: modeling, forecasting and impacts on revenue management*. PhD thesis, Massachusetts Institute of Technology, 2016.

US DOT. Bumping & oversales. https://www.transportation.gov/individuals/aviation-consumer-protection/bumping-oversales, 2020. Accessed: 2020-05-03.

Peter P Belobaba. Optimization models in rm systems: Optimality versus revenue gains. *Journal of Revenue and Pricing Management*, 15(3-4):229–235, 2016.

Peter P Belobaba, William G Brunger, and Michael D Wittman. Advances in airline pricing, revenue management, and distribution: Implications for the airline industry. https://www.transportation.gov/individuals/aviation-consumer-protection/bumping-oversales, 2017. Accessed: 2020-05-06.