Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis in Business Analytics & Quantitative Marketing

Econometrics and Management Science

Deep Multi-Agent Interaction in Airline Revenue Management

Supervisor:
Prof. dr. S.I. Birbil
Second assessor:
Prof. dr. S. Sharif Azadeh

Authors:
Trevor Visser (443480)

January 26, 2021

Abstract

In revenue management problems, sophisticated solutions are generally based on approximations of reality. To provide a more flexible approximation, we explore a single-leg airline revenue management problem from a multi-agent reinforcement learning (MARL) perspective. We create a MARL environment that simulates the interaction between a single airline agent and multiple customer agents with their own respective goals. Thereafter, we optimize both types of agents to analyze the resulting behaviors. From the resulting strategies, we discover that the optimized agents exhibit human-like intelligence and an ability to adapt to varying simulation scenarios. Moreover, the resulting strategies—which include catering to specific types of customers and sampling expected demand—exhibit strong similarities to what one might expect from reality given similar environment circumstances. From these results, we then further discuss the potential risks and rewards that MARL can bring to other fields from both a research and business point of view.

Contents

1 Introduction

With the increasing prevalence of innovations in machine learning and artificial intelligence, areas of nonlinear optimization and resource management previously untouched have become readily available. Chief among advancements in artificial intelligence is in the form of reinforcement learning (RL). Generally speaking, RL is an approximation of traditional dynamic programming (DP) techniques. While DP has been seen to provide excellent results in practice (R. Bellman, 1966; Ross, 2014; R. E. Bellman and Dreyfus, 2015), its actual implementation is highly restricted to domains where complete information is available—namely complete knowledge of event probabilities and model formulation. RL, on the other hand, circumvents these restrictions by providing a methodology that learns the event probabilities and model formulation without any prior knowledge—thus minimizing human-based input. As explored by Kaelbling, Littman, and Moore (1996), RL is typically formulated as a Markov decision process (MDP) allowing for standard DP approximations. However, only with the proposed inclusion of neural networks, (deep) RL has garnered significant attention and results. With the addition of neural networks, RL's ability to approximate non-linear functions associated with optimizing their specified goals (especially in MDPs with large state spaces where the curse of dimensionality reigns supreme) has greatly improved (Mnih, Kavukcuoglu, Silver, Graves, et al., 2013, Mnih, Kavukcuoglu, Silver, Rusu, et al., 2015).

Because of its general format, RL has found application in many fields. Of particular interest is in the field of revenue management. Revenue management is essentially the process of optimizing strategies that maximize revenue growth for any particular sector. These strategies typically include decisions of whether to buy or sell, and in what quantities. Historically, these problems have often been solved through DP techniques. However, due to limitations such as problem specific formulation and infeasibility in large state spaces, the network management environments typically remained small and thus were only applicable to reality under specific conditions. Recently, with the influx of RL algorithms, research has begun to shift towards RL applications in revenue management problems with mostly positive results (Rana and Oliveira, 2015; Shihab et al., 2019).

However, amongst these RL implementations, an inherent limitation is persistent in the singular number of agents/strategies being optimized. For example, in a simplified version of the airline revenue management problem, a single airline agent must learn how to allocate seats amongst customers whom randomly arrive and cancel their tickets according to pre-specified distributions. Although an optimal strategy can be found, a limitation is inherent in the pre-coding of the customer behavior—applicability to real life. As customers in the simulation environment act according to pre-specified distributions, the practicability of the optimal strategy is thus based on the strict assumption that the distributions themselves moderately reflect real life. Often, this assumption is not met and if it is, the distributions then become problem specific and fail to incorporate human-centric adaptability. Multi-agent RL (MARL) can potentially overcome this limitation.

In MARL, *multiple* agents are put in a simulation environment where they all jointly optimize their respective goals through interactions with the environment and each other. With regards to the airline problem, MARL can potentially allow for multiple airlines (simulating competition) and multiple customers who are all optimizing their respective strategies. This provides a more realistic simulation scenario as agents in any trade-off situation are continuously learning. As an example of the potential effectiveness of multi-agent interaction, a game of hide and seek was simulated by OpenAI where multiple agents with different goals were pitted against one another and could only learn from their own experiences (B. Baker et al., 2019). The resulting strategies developed were similar to what one would expect from humans but eventually continued into realms that defied expected creativity (box-surfing and finding loopholes in the simulation environment). This unexpected creativity and ability to converge to solutions despite the added state space from multiple agents has been seen in multiple papers for both competitive and cooperative environments (Lowe et al., 2017; Littman, 1994; Gupta and Dukkipati, 2019). From the results of these papers, a particular highlight is in the evolution of dynamic behavior between agents—a potential solution to the rigidity and static nature of pre-coded behavior. In the ideal scenario where MARL is able to adequately approximate reality, it becomes possible to not only find optimal strategies given current data, but to also explore hypothetical situations where no data is available.

From these considerations, this thesis explores a MARL application in revenue management problems—particularly in a single-leg airline revenue management problem. We provide the following contributions: (1) a complete MARL implementation of the single-leg airline revenue management problem with optimization details; (2) empirical results that demonstrate how MARL provides intelligent agents capable of adaptation and achieving respectable performance; (3) an analysis into how well the MARL agents are able to approximate what one may expect from reality.

2 Related Work

Although MARL can be considered a fairly new research topic, multi-agent settings have long been a field of interest. For example, Sims (1994) and Yaeger (1994) both consider the evolution of behavior through the interaction of multiple agents. In the former, a genetic algorithm was used for the optimization and their research was focused on emerging behavior complexities whereas the latter creates a multi-agent environment meant to mimic reality from a biological point of view. More recently, Wang et al. (2019) explore the generation of complex behaviors while subsequently altering the simulating environment to assess overall agent adaptability. From the perspective of RL, the most well-known multi-agent application is AlphaGo where an agent learned to play the game of Go at a superhuman level from only playing against itself (Silver et al., 2017). From here, further advancements were seen with OpenAI (2018) and Vinyals et al. (2019) where agents were trained to perform extremely well in the multiplayer games of Dota and Starcraft respectively. A key difference between AlphaGo and the algorithms used for Dota and Starcraft is that AlphaGo dealt with complete information environments whereas Dota and Starcraft dealt with incomplete information. Thus their algorithms signified an important change in the approach of optimizing agents in partially observable environments.

Revenue management is a thoroughly researched topic with varying degrees of applications such as in the hospitality industry, delivery services, and airline industry. Despite the large spread, Talluri and Van Ryzin (2006) provide one of the first general revenue management overviews. In their book they consider a broad range of revenue management applications in multiple industries and unify it all to form a revenue management reference source. From the perspective of airline revenue management, there are several dynamic programming approaches to settings ranging from simplified single-leg problems to more complex network-based problems. Regarding the single-leg applications, Otero and Akhavan-Tabatabaei (2015) propose a stochastic pricing model that prices tickets depending on the expected inter-arrival probabilities; Aydın et al. (2013) explore the computation of overbooking limits with two models for multiple airline classes; and D. Zhang and Cooper (2009) additionally look at stochastic pricing but instead emphasize optimization heuristics under high-dimensionality. For the complex network-based problems, both Bertsimas and De Boer (2005) and Birbil et al. (2014) tackle the computation of overbooking limits by decomposing the network problem into multiple single-leg parts. Thereafter, however, the two diverge such that Bertsimas and De Boer (2005) solve for the booking limits through a hybrid stochastic gradient descent algorithm whereas Birbil et al. (2014) make use of mathematical programming techniques. As an approximation to the restrictions in dynamic programming techniques, RL applications in revenue management have become more common as well. For example, Gosavii, Bandla, and Das (2002) explore overbooking in airline revenue management with RL, Rana and Oliveira (2015) consider how to price products over time, and Gosavi, Ozkaya, and Kahraman (2007) tackles model-free seat allocation in discrete-event simulator.

Additionally, there has been research done into multi-agent approaches for revenue management. Lin et al. (2018) explore a multi-agent reinforcement learning that attempts to maximize revenue and customer satisfaction through managing the transportation of resources. In this case, the agents cooperate with one another. Similarly, Bazzan (2009) consider a multi-agent application in traffic control where several agents work together to provide efficient traffic control and minimize any congestion. While not directly associated with increasing revenue, the efficiency gain is nonetheless akin to revenue gain in other applications.

3 Background

In this section, we outline a few necessary prerequisites that we assume to be known throughout the thesis. These include the three most commonly studied forms of game theory: Markov decision processes, Markov games, and extensive-form games (discussed using the notation found in K. Zhang, Yang, and Başar, 2019). In addition to explaining these different game forms, we also briefly discuss the reinforcement learning optimization algorithm Soft Actor-

Critic (Haarnoja et al., 2018).

3.1 Markov Decision Process

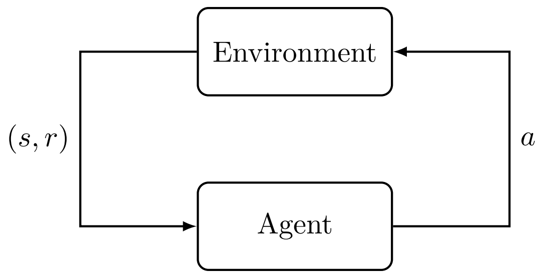


Figure 1: Markov Decision Process
A schematic diagram displaying the general system of a Markov decision process (MDP). As a single-agent system, the agent receives states s from the environment, sends an action a in response, and subsequently receives a new state and reward (s, r).

Assume we have a simulation environment where a single customer agent seeks to buy a ticket from an airline. At each time step in the simulation, the sole customer agent observes the price of the ticket and decides if it wants to purchase the ticket or wait—once the customer purchases a ticket, the simulation ends. Naturally, the customer is rewarded for purchasing the airline ticket at a lower price. This type of simulation is an exmaple of a Markov Decision Process (MDP). MDPs denote an environment characterized by a set of states $\mathcal{S}$ and actions $\mathcal{A}$. To transition from state $s_t \in \mathcal{S}$ to $s_{t+1} \in \mathcal{S}$, an action $a_t \in \mathcal{A}$ must be taken. We represent this state transition probability as $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Upon transition, a reward is given as a function of the state and chosen action $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Referring back to our example, the state $s_t \in \mathcal{S}$ would consist of the ticket price and the action $a_t \in \mathcal{A}$ would be to either buy or wait. In general, the agent chooses actions through some stochastic policy $\pi : \mathcal{S} \to \mathcal{A}$. Thus its objective—as written in K. Zhang, Yang, and Başar (2019)—is to find a policy π that chooses actions $a_t \sim \pi(\cdot|s_t)$ such that the following discounted total expected reward function is maximized:

$$\mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r_t \Big| a_t \sim \pi(\cdot|s_t), s_0\Big], \tag{1}$$

where $\gamma \in [0, 1]$ is a tunable discounting factor that alters how forward looking the agent is. Combining these characteristics, MDPs are summarized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$. From the discounted total reward function, one can further specify a *state-action value* (commonly denoted as a Q-function) function

$$Q_\pi(s, a) = \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r_t \Big| a_t \sim \pi(\cdot|s_t), a_0 = a, s_0 = s\Big] \tag{2}$$

As can be seen, the Q-function essentially estimates the total expected discounted reward if starting from state $s_0 = s \in \mathcal{S}$ with action $a_0 = a \in \mathcal{A}$..

3.2 Markov Game

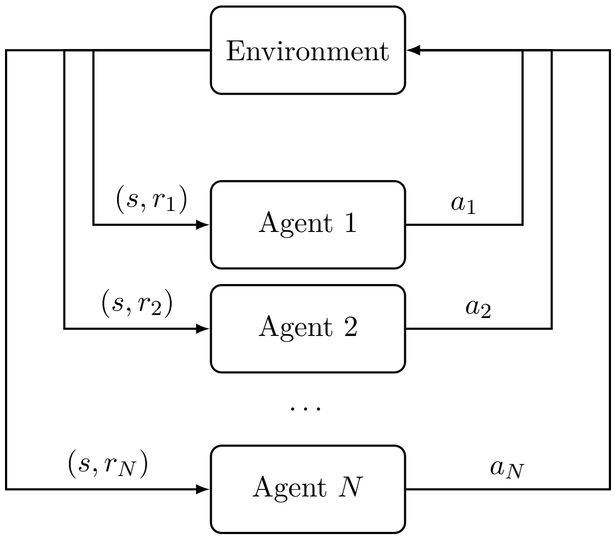


Figure 2: Markov Game
A schematic diagram displaying the general system of a Markov game (MG). In a MG, N agents receive the state s from the environment and respond with actions a_i simultaneously. Once these actions are received, the environment provides each agent with their own reward r_i and the following state s.

Let us assume that now instead of a single customer agent, we have $N > 1$ customer agents—each wanting to purchase an airline ticket. At every time step, each customer agent observes the airline ticket price and decides whether to buy or wait. This multi-agent extension is otherwise known as a Markov game (MG). As implied, a MG is a generalization of MDPs that allows for a set of $N > 1$ agents, $\mathcal{N} = \{1, \dots, N\}$. Similarly, a MG can be characterized by a tuple: $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \mathcal{T}, \{r_i\}_{i\in\mathcal{N}}, \gamma)$. The environment is characterized by a set of states $\mathcal{S}$. We consider the application of partially observable Markov games where each agent can only partially observe the full state. For example, this means that one customer agent can not observe how much capital another customer agent has. As a result, each agent has its own set of observations $\{\mathcal{O}_i\}_{i=1}^N$ and actions $\{\mathcal{A}_i\}_{i=1}^N$ such that $\mathcal{O}_i$ is computed through some function $o_i : \mathcal{S} \to \mathcal{O}_i$ and actions are chosen through some agent-specific stochastic policy $\pi_i : \mathcal{O}_i \to \mathcal{A}_i$. In a MG, all agents act simultaneously and thus the transition from one state to another is dependent on the joint action choices: $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \ldots \times \mathcal{A}_N \to \mathcal{S}$. Rewards subsequently become agent specific as $r_i : \mathcal{S} \times \mathcal{A}_i \to \mathbb{R}$ with each agent attempting to maximize their own total discounted expected reward

$$\mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r_{i,t} \Big| a_{i,t} \sim \pi_i(\cdot|s_t), s_0\Big].$$

Within a multi-agent setting, there naturally exists a Q-function counterpart for each agent i.It is defined to be

$$Q_{\pi_i,\pi_{-i}}(s, a) = \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r_{i,t} \Big| a_{i,t} \sim \pi_i(\cdot|s_t), a_0 = a, s_0 = s\Big]$$

where we set $-i$ to denote the agents in the set $\mathcal{N} \setminus \{i\}$. In a MG where all agents act simultaneously, it is important to note that the Q-function is dependent on the joint policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ where $\pi(a|s) := \prod_{i \in \mathcal{N}} \pi_i(a_{i,t}|s_t)$ as opposed to just a single agent's actions.

3.3 Extensive-Form Game

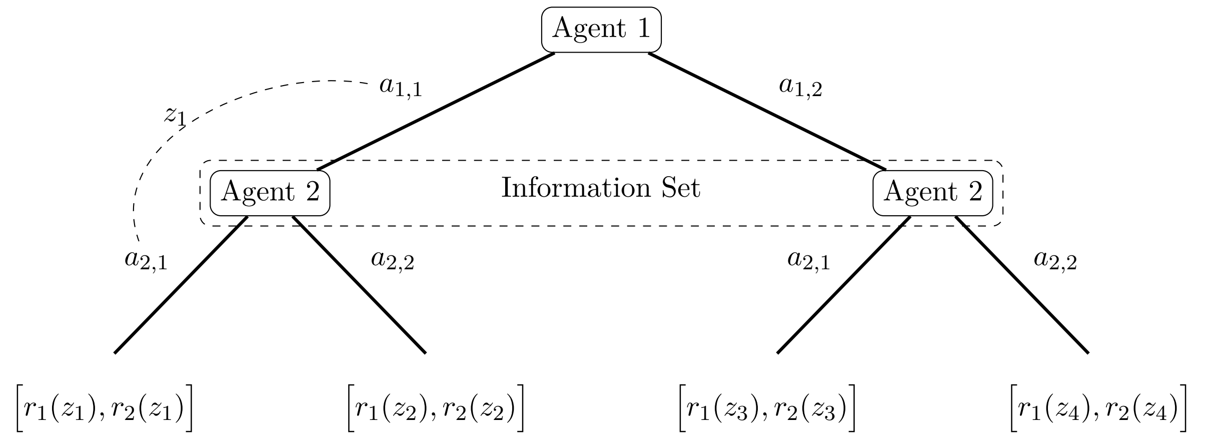

Agent 1
$a_{1,1}$
$a_{1,2}$
z_1
Agent 2
Information Set
Agent 2
$a_{2,1}$
$a_{2,2}$
$a_{2,1}$
$a_{2,2}$
$\left[r_1(z_1), r_2(z_1)\right]$
$\left[r_1(z_2), r_2(z_2)\right]$
$\left[r_1(z_3), r_2(z_3)\right]$
$\left[r_1(z_4), r_2(z_4)\right]$

Figure 3: Extensive-Form Game
A schematic diagram displaying the general system of a two-player extensive-form game (EFG). In this EFG, agents act in a sequential fashion such that agent 1 chooses an action and agent 2 then chooses another action in response generating a terminal history z and thus agent-specific rewards $r_i(z)$. An EFG can be separated into perfect/imperfect information cases. In the perfect information case, agent 2 will have full information on which node they find themselves in (thus having knowledge on what action agent 1 has taken). Imperfect information, as displayed in this diagram, depicts the scenario where agent 2 has no knowledge of agent 1's actions and thus is unable to differentiate where it finds itself in the information set.

In the airline examples presented for the MDP and MG subsection, we assume that the airline ticket prices are known in at each time step. Let us now consider a simulation where two types of agents interact: a single airline agent that sets prices, and a single customer agent that wants to purchase a ticket from the airline. Thus, in each time period, the airline first sets a price and then the customer observes the price to decide to buy or wait. This type of sequential interaction is an example of an extensive-form game (EFG). Unlike in a MG where agents act simultaneously, an EFG considers the premise where agents act sequentially.

EFGs are defined by the tuple $(\mathcal{N} \cup \{c\}, \mathcal{H}, \mathcal{Z}, \mathcal{A}, \{r_i\}_{i \in \mathcal{N}}, \tau, \pi_c, \mathcal{S})$ where $\mathcal{N} \cup \{c\}$ is the set of $N > 1$ agents with an additional chance agent c. The chance agent c possesses a fixed stochastic policy π_{chance} that specifies how the environment is randomized. From the tuple, $\mathcal{H}$ and $\mathcal{A}$ denotes the set of all possible histories and actions respectively where a history $h \in \mathcal{H}$ is a sequence of actions taken between a start and end point of the game. From this, we define $\mathcal{Z} \subseteq \mathcal{H}$ to be a subset of histories that end at the completion of the game. Let $\mathcal{A}(h)$ be the set of all possible actions after a non-terminal history h such that if an agent takes action $a \in \mathcal{A}(h)$, we transition to a new history $h' \in \mathcal{H}$. If h' leads to the end of the game ($h' \in \mathcal{Z}$), it is terminal and a reward is given to each agent. Subsequently, the reward function is defined to be $r_i : \mathcal{Z} \rightarrow \mathbb{R}$. Furthermore, τ is defined to be the identification function $\tau : \mathcal{H} \rightarrow \mathcal{N} \cup \{c\}$, which determines what agent can act at each history. Naturally, if $\tau(h) = c$, the chance agent acts.

Lastly, $\mathcal{S}$ is the set of information states that partition $\mathcal{H}$: if we let $s \in \mathcal{S}$ be an information state and $h_1, \ldots, h_j \in \mathcal{H}$ are different histories that result in s (in other words, $h_1, \ldots, h_j \in s$), then it holds that $\tau(h_1) = \ldots = \tau(h_j)$ and $\mathcal{A}(h_1) = \ldots = \mathcal{A}(h_j)$. This ultimately means that given a specific information state s, agents are unable to differentiate between the histories that resulted in s.

Logically, since $\tau(h) = \tau(h')$ and $\mathcal{A}(h) = \mathcal{A}(h')$ for $\forall h, h' \in s$, it is sufficient to set $\mathcal{A}(h)$ as $\mathcal{A}(s)$ and $\tau(h)$ as $\tau(s)$. From this connection, agents are able to map each information state to an action through a stochastic policy $\pi_i : \mathcal{S}_i \to \mathcal{A}(s)$ where $\mathcal{S}_i = \{s \in \mathcal{S} : \tau(s) = i\}$. If one denotes $\pi = (\pi_1, \ldots, \pi_N)$ as the joint policy of all N agents and $\mathcal{I} : \mathcal{H} \to \mathcal{S}$ as the mapping between the histories and information sets, the probability of reaching history $h \in \mathcal{H}$ becomes

$$\eta_\pi(h) = \prod_{h':h' \sqsubseteq h} \pi_{\tau(h')}(a|\mathcal{I}(h')) = \prod_{i \in \mathcal{N} \cup \{c\}} \Big(\prod_{\substack{h':h' \sqsubseteq h, \\ \tau(h')=i}} \pi_i(a|\mathcal{I}(h')) \Big) \tag{3}$$

where $h' \sqsubseteq h$ denotes that h' is the prefix of h, meaning that history h can be reached from h' through some sequence of actions. From Equation 3, the goal of each agent is thus to maximize their own expected reward

$$R_i(\pi) = \sum_{z \in \mathcal{Z}} \eta_\pi(z) r_i(z). \tag{4}$$

3.4 Soft Actor-Critic

In order to optimize a stochastic policy, gradients must be calculated on the inherently random nature of the policy output. As a result, a common problem in the optimization of stochastic policies is high variance and subsequent instability. To combat this, other optimization algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) apply a form of smoothing to provide better gradient estimates. Soft Actor-Critic (SAC) is a relatively new reinforcement learning (RL) algorithm from Haarnoja et al. (2018) that proposes a method to find the optimal policy π^*. In addition to possessing the smoothing feature present in PPO, it additionally includes a new type of objective function that prevents the algorithm from converging to a bad local optimum while simultaneously speeding up the learning process. In their algorithm, they define the policy π to be a neural network parameterized by θ to map a state to an action distribution. Following the notation of subsection 3.1, they augment the RL objective of Equation 1 with expected entropy:

$$\pi_\theta^* = \underset{\theta}{\operatorname{argmax}} \, \mathbb{E}\Big[\sum_{t \geq 0} \gamma^t r_t + \alpha \mathcal{E}[\pi_\theta(\cdot|s_t)] \Big| a_t \sim \pi_\theta(\cdot|s_t), s_0\Big], \tag{5}$$

where $\mathcal{E}[\pi_\theta(\cdot|s_t)] = -\log \pi(\cdot|s_t)$ is the entropy of π_θ at state s_t and α is a temperature parameter that weighs the relative important between the reward and entropy. The algorithm additionally makes use of another neural network, Q_ϕ, to approximate the Q-function of Equation 2. To perform SAC, π_θ acts in the simulation environment to collect tuples (s_t, a_t, r_t, s_{t+1}) and store them in some trajectory storage $\mathcal{D}$. From this trajectory, batches of tuples are randomly sampled to update ϕ, θ, and α by minimizing the following loss functions through stochastic gradient descent:

$$\min_{\phi} \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\Big[\frac{1}{2}(Q_\phi(s_t,a_t) - y(s_t,a_t))^2\Big] \tag{6}$$

where $y(s_t,a_t) = r(s_t,a_t) + \gamma \mathbb{E}_{\substack{s_{t+1}\sim\mathcal{T}(s_t,a_t)\\ a_{t+1}\sim\pi_\theta(s_{t+1})}}\Big[Q^{targ}(s_{t+1},a_{t+1}) - \alpha \log \pi_\theta(a_{t+1}|s_{t+1})\Big]$. Q^{targ} denotes another Q-function neural network whose parameters are set to periodically evolve to become the parameters of Q_ϕ;

$$\min_{\theta} \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\Big[\alpha \log \pi_\theta(a_t|s_t) - Q_\phi(s_t,a_t)\Big]; \tag{7}$$

and

$$\min_{\alpha} \mathbb{E}_{a_t\sim\pi_\theta}\Big[-\alpha(\log \pi_\theta(a_t|s_t) + H)\Big] \tag{8}$$

where H is some pre-defined entropy target. Haarnoja et al. (2018) additionally make use of the clipped Q-function method (Fujimoto, Van Hoof, and Meger, 2018) which trains two separate Q-function networks (ϕ_1,ϕ_2) and takes the minimum of their outputs such that $Q_\phi(s_t,a_t) = \min(Q_{\phi_1}(s_t,a_t), Q_{\phi_2}(s_t,a_t))$.

4 Airline Environment

To understand the results and highlight their key features, we use this section to describe our multi-agent airline environment. We begin with a qualitative description of the environment that describes the agent interaction dynamics and the timestep progression. From here, we then move to a more formal description of the environment as a hybrid between a Markov and extensive-form game.

4.1 The Simulation

In order to perform an analysis into the effectiveness of MARL, we create a 100[1] timestep simulator based on airline revenue management. In this simulation, two different kinds of agents interact with one another with opposing goals: airlines that sell tickets and seek to maximize their end profit, and customers who subsequently buy the aforementioned tickets and seek to minimize their costs and maximize their utility. We consider the controlled scenario where a single airline agent offers tickets for a single-leg flight to multiple customers. For this flight, the airline is given a fixed number of tickets with different ticket classes (2 first class tickets, 3 second class tickets, and 5 third class tickets resulting in a total flight capacity of 10). These ticket classes differentiate from one another in the sense that first class is of a higher quality than second class, which in turn is of a higher quality than third class. In reality, a higher quality flight class consists of more benefits than lower class tickets. Naturally, the airline has fixed investment costs to provide these benefits and thus we define each ticket class to have a fixed investment cost in the beginning of the simulation period. This added feature asserts that for an airline to have positive profit, they must generate enough revenue to offset the initial

[1]Although we choose a 100 timestep simulation, the resulting strategies of the agents are independent of the length of the simulation due to the input scaling discussed in Subsection 4.2.2.

ticket investment costs.

For every time period until flight departure, the airline agent is tasked with determining the prices and booking limits for each ticket class. Each customer that buys a ticket has a chance of not showing up for the flight and/or cancelling their purchased ticket. Thus the booking limits are included such that an airline agent may decide to overbook or even not sell a specific ticket class. For example, at a specific time period, the airline may choose to set prices of 0.8, 0.7, and 0.6 for each ticket class respectively and additionally decides to only sell 1 out of the 2 first class tickets, all 3 second class tickets, and 6 third class tickets. However, in the event that there are too many customers at the time of departure, the airline must bump customers off the flight and subsequently incurs a bump cost relative to the price of each ticket sold in excess. Customers, on the other hand, are given an initial amount of capital to buy tickets at the prices that the airline agent sets—provided the ticket in question is available. After the airline has set their prices and booking limits, each customer makes the decision to either buy a ticket from a certain ticket class or to do nothing. It is important to note that not every ticket is available to be bought in each time period: if the airline decides to not sell a specific ticket (by setting the booking limit appropriately) or a specific customer does not have enough capital, the ticket in question becomes unavailable to that customer. Our simulation adheres to the constraint that a customer can only hold one ticket—if a customer decides to purchase another ticket while already having one, their original ticket is first cancelled. Customers are additionally randomly initialized to be of three income brackets: lower, middle, or upper class. Depending on their income bracket, the starting capital and the probability of not showing up for the flight at departure changes.

We place key importance on the interaction order between airline and customer agents. At the beginning of each new time period, the airline agents act first to set the price and booking limits. Once this is done, all customer agents receive the newly updated prices and act simultaneously. After the customers act, the airline receives the results of the customers' actions (tickets bought/cancelled, revenue generated/lost) and uses this information to set new price and booking limits restarting the interaction cycle. As a result of this construction, both airline and customer-specific rewards are only received after the customers act.

4.2 Markov/Extensive-Form Environment

The simulation described in subsection 4.1 follows neither a MG or EFG format but is rather a unification of the two types: it is a perfect information repeated game (a specific type of EFG) where the histories consider series of *joint* actions (MG simultaneity). For further explanation, we refer back to the EFG notation of subsection 3.3. As its name implies, a repeated game is essentially an EFG that repeats itself finitely and perfect information states that, for an information set $\forall s \in \mathcal{S}$, $|s| = 1$.

In each time period t of the simulation, an EFG$_t$ is played out between the airline and customer agents. The airline (denoted by subscript a) observes their information state $s_{t,a} \in \mathcal{S}$ ($|s_{t,a}| = 1$)

and outputs the joint action $\mathbf{a}_{t,a} = [p_{1,t}, \dots, p_{j,t}, b_{1,t}, \dots, b_{j,t}]^T \in \mathcal{A}(s_{t,a})$ where p_j and b_j denote the price and booking limit for ticket class j respectively. The chosen action $\mathbf{a}_{t,a}$ results in history $h_t \in \mathcal{H}$. The C customer agents (denoted by subscript c) subsequently observe information set $s_{t,c} \in \mathcal{S}$ such that $\mathcal{I}(\mathbf{a}_{t,a}) = \mathcal{I}(h_t) = s_{t,c}$ which comprises of customer-specific observations and knowledge on what action the airline has taken. They then each choose an action x_i forming the joint action $\mathbf{a}_{t,c} = [x_{1,t}, \dots, x_{C,t}]^T \in \mathcal{A}(s_{t,c})$ which generates a new history $h'_t \in \mathcal{H}$. The history of h'_t comprises of both $\mathbf{a}_{t,a}$ and $\mathbf{a}_{t,c}$ which completes the EFG$_t$. Thus $h'_t := z_t \in \mathcal{Z}$ and a reward is generated for the airline $r_{t,a}(s_{t,a}, s_{t,c}, z_t)$ and customers $r_{i,t,c}(s_{t,a}, s_{t,c}, z_t)$. For notational convenience, we summarize these rewards as $r_{t,a}(z_t)$ and $r_{i,t,c}(z_t)$ respectively. This EFG repeats itself each time period t until the flight departure at time period $T = 100$. To better visualize this process, we provide Figures 4a and 4b where the former gives the environment logic from a MG perspective while the latter gives the environment logic from an EFG perspective. We additionally provide the environment pseudocode in Algorithm 1.

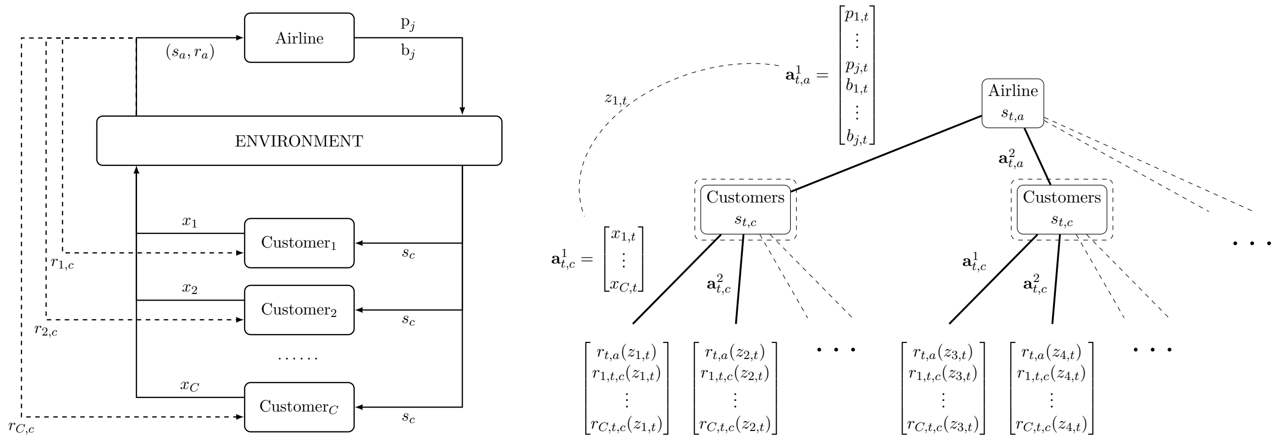
Algorithm 1 Airline Multi-Agent Simulation

```
 Input: airline strategy $\pi_a$, customer strategies $\pi_{i,c}$ for $i = 1, \dots, C$ customers;
 1. Randomize environment settings using chance agent policy $\pi_{chance}$;
 2. Set time period $t \leftarrow 0$;
 3. Set agent $\leftarrow$ airline;
 4. Set $\mathrm{p}_{j,t} = \mathrm{b}_{j,t} \leftarrow 0$ for $j = 1, \dots, J$ ticket classes;
 5. repeat
 6.     if agent = airline do
 7.         Get $\mathbf{a}_{t,a} = [p_{1,t}, \dots, p_{j,t}, b_{1,t}, \dots, b_{j,t}]^T$ from $\pi_a$ using information state $s_{t,a} \in \mathcal{S}$;
 8.         Set environment prices/booking limits as $p_{j,t}$ and $b_{j,t}$;
 9.         Get new history $h_t \in \mathcal{H} \setminus \mathcal{Z}$ and information state $s_{t,c} = \mathcal{I}(\mathbf{a}_{t,a}) = \mathcal{I}(h_t)$;
10.         Set agent $= \tau(h_t) \leftarrow$ customer;
11.     end if
12.     if agent = customer do
13.         Get joint action $\mathbf{a}_{t,c} = [x_{1,t}, \dots, x_{C,t}]^T$ from $\pi_{i,c}$ using information state $s_{t,c} \in \mathcal{S}$;
14.         Process customer $i$'s action in environment;
15.         Get new history $z_t \in \mathcal{Z}$ and information state $s_{t,a} = \mathcal{I}(z_t)$;
16.         Get airline and customer rewards $[r_{t,a}(z_t), r_{1,t,c}(z_t), \dots, r_{C,t,c}(z_t)]^T$;
17.         Set agent $= \tau(z_t) \leftarrow$ airline;
18.         Set $t \leftarrow t + 1$;
19.     end if
20. until $t = T$
```

Because there are two differing types of agents, the set of information states $\mathcal{S}$ can be partitioned into disjoint airline $\mathcal{S}_a$ and customer $\mathcal{S}_c$ parts such that $\mathcal{S} := \mathcal{S}_a \cup \mathcal{S}_c$. The partitioning can be extended further when we consider that the simulation itself is a time-dependent repeated game. It was earlier mentioned that the actions available for customers to take is dependent on several factors such as available capital and booking limits. This ultimately means that the actions available in the EFG$_t$, $\mathcal{A}(h_t)$ $h_t \in \mathcal{H}$, is dependent on the outcomes of EFG$_{t-1}, \dots,$ EFG$_0$. Thus, we can partition $\mathcal{S}$ such that $\mathcal{S} := \bigcup_{l \in \{a,c\}} \bigcup_{t=0}^{T} \mathcal{S}_{t,l}$. By definition, this partition then holds for the action space $\mathcal{A}(s)$ depending on the $s \in \mathcal{S}$. For the sake of brevity, we will henceforth denote $s_{t,a}$, $s_{t,c}$, $\mathbf{a}_{t,a}$, and $\mathbf{a}_{t,c}$ to be arbitrary information and action states from $\mathcal{S}_{t,a}$, $\mathcal{S}_{t,c}$, $\mathcal{A}(s_{t,a})$, and

$\mathcal{A}(s_{t,c})$ respectively. With regards to the identification function τ, the choice between airline and customer agents is trivial as the two act after one another. Since EFG$_t$ ends after the customers act and a reward is given, we naturally can state that the terminal histories $z_t \in \mathcal{Z}$ result only from the joint customer action. In other words, $\tau(z_t) =$ airline and $\tau(h_t) =$ customer if $h_t \in \mathcal{H} \setminus \mathcal{Z}$.

With this general overview explained, we now turn to the implementation details of the chance agent, the information states, the action space, and the reward functions for both the airline and customer agents. To maintain a simplified environment space, we bound the prices, costs, and spending money to be between 0 and 1. Throughout this subsection, we additionally list our chosen simulation parameters. However, for a more concise view of these parameters, please refer to Appendix A.



Airline
p_j
b_j
(s_a, r_a)
ENVIRONMENT
x_1
Customer$_1$
s_c
$r_{1,c}$
x_2
Customer$_2$
s_c
$r_{2,c}$
......
x_C
Customer$_C$
s_c
$r_{C,c}$
$\mathbf{a}^1_{t,a} = \begin{bmatrix} p_{1,t} \\ \vdots \\ p_{j,t} \\ b_{1,t} \\ \vdots \\ b_{j,t} \end{bmatrix}$
$z_{1,t}$
Airline $s_{t,a}$
$\mathbf{a}^2_{t,a}$
Customers $s_{t,c}$
Customers $s_{t,c}$
$\mathbf{a}^1_{t,c} = \begin{bmatrix} x_{1,t} \\ \vdots \\ x_{C,t} \end{bmatrix}$
$\mathbf{a}^2_{t,c}$
$\mathbf{a}^1_{t,c}$
$\mathbf{a}^2_{t,c}$
$\begin{bmatrix} r_{t,a}(z_{1,t}) \\ r_{1,t,c}(z_{1,t}) \\ \vdots \\ r_{C,t,c}(z_{1,t}) \end{bmatrix}$
$\begin{bmatrix} r_{t,a}(z_{2,t}) \\ r_{1,t,c}(z_{2,t}) \\ \vdots \\ r_{C,t,c}(z_{2,t}) \end{bmatrix}$
$\begin{bmatrix} r_{t,a}(z_{3,t}) \\ r_{1,t,c}(z_{3,t}) \\ \vdots \\ r_{C,t,c}(z_{3,t}) \end{bmatrix}$
$\begin{bmatrix} r_{t,a}(z_{4,t}) \\ r_{1,t,c}(z_{4,t}) \\ \vdots \\ r_{C,t,c}(z_{4,t}) \end{bmatrix}$

(a) Markov Game **(b)** Timestep t Extensive-Form Game

Figure 4: Simulation Logic
This figure provides a schematic diagram of the airline environment logic from both a Markov and extensive-form game perspective. If we consider 4a, we see that the environment provides the airline agent with its specific information state s_a. Once received, the airline provides the environment with its action—namely the price p_j and booking limit b_j for each ticket class j. The environment receives this action and subsequently alters the customer information states s_c such that they reflect the newly set prices p_j. The C customers then receive their information states s_c and send their chosen actions x_i back to the environment. Once the environment receives the customers' actions, it updates s_a with the outcomes of the customers' actions and returns rewards for both the airline r_a and customers $r_{i,c}$ as well as the new s_a to the airline (beginning a new time period). This interaction repeats itself until the simulation period is over. In 4b, we display the aforementioned interaction of a single time period t in an EFG format. The airline agent receives its information state $s_{t,a}$, chooses an action $\mathbf{a}_{t,a}$, and generates the customer information state $s_{t,c}$. The customers observe $s_{t,c}$, choose their actions $x_{i,c}$, and thus generate the rewards $r_{t,a}$ and $r_{i,t,c}$ for airlines and customers respectively.

4.2.1 The Chance Agent

Of key importance in the simulation is messiness in the form of randomization. Reality is often subject to a variety of individuals with different tastes and circumstances—many of which play a role in their decision making. To account for this, we make use of the chance agent c and its policy π_{chance} to provide stochastic randomization at each initialization of the environment. In repeated games, it is common for the chance agent to act in each new EFG. However, as we consider time dependent repeated games (such that the results of EFG$_t$ depend on $\text{EFG}_{t-1}, \ldots, \text{EFG}_0$) our chance agent acts only in EFG_0.

Of the randomizations, the most important detail is that the number of customers is not constant. The chance agent uniformly samples between having 2 to 20 customers. Not only does this feature allow for realistic situations where the airline agent must deal with varying demand, it additionally allows for an analysis into situations where supply exceeds demand, supply meets demand, or demand exceeds supply. Although the customers act simultaneously, the simulation adheres to a random ordering upon each initialization such that customers ranked first will have priority in buying tickets over others and so on. This random ordering, in conjunction with the booking limits set by the airline, adds an extra degree of complication for the customers. For example, say that an airline has sold zero tickets and decides to sell a maximum of two second class tickets in the next time period t. At time period t, all C customers will be allowed to make the decision to purchase a second class ticket but that does mean each customer will receive one. Customer i (who is ranked ith in purchasing order) will only successfully purchase a second class ticket if, of the remaining customers $i = 1, 2, \ldots, i-1$, none or at most one decide to similarly purchase a second class ticket. Thus, in order to maximize their chances of purchasing a ticket, customers must learn to consider demand and supply in their choices. This random ordering also applies to bumping passengers—the customer ranked 1st will be last to be bumped from the flight as opposed to the customer ranked last. For the initialization of income brackets, the chance agent defines that approximately 30% of the customers are lower class, 60% are middle class, and 10% are upper class. Starting capital is uniformly sampled between two bounds depending on the respective income bracket—lower: $[0.25, 0.5]$, middle: $(0.5, 0.75]$, upper: $(0.75, 1.0]$. Furthermore, the chance agent also sets that the probability of a customer not showing up to the flight is dependent on their income bracket—lower: 10%, middle: 10%, upper: 20%. Although these simulation parameters were chosen based on our intuition and are not necessarily backed by data/research, there values are sufficient to achieve the purpose of our research.

4.2.2 Information State

In a typical perfect information EFG, an information state $\forall s \in \mathcal{S}$ would consist of all environment information as well as what action history led to s and in an idealized world, this perfect information would hold. However, this is not the case in reality as customers would not have real-time access to information such as the revenue/expenditure of airlines and airlines would not have access to the amount of wealth that each customer has. To account for this imperfect information access, our simulation adheres to the partially observable MG format with sets of agent-specific observations—previously defined in subsection 3.2 by the function $o_i : \mathcal{S} \to \mathcal{O}_i$. However, in our application, $\mathcal{S}$ denotes the set of information states rather than just normal MG states. Thus when an agent observes either information state $s_{t,a}$ or $s_{t,c}$, we define that they actually only observe the observation vectors $o_i(s_{t,a})$ and $o_i(s_{t,c})$ respectively. We otherwise denote these observations as $o_{t,a}$ and $o_{i,t,c}$ suppressing the i for airlines as we only consider one airline[2].

The observation vector for airlines $o_{t,a}$ consists of the following information: (1) the current timestep t scaled by the total timesteps before departure T; (2) gross profit (gross revenue - gross expenditure); (3) the current number of available seats scaled by the total number of seats available per ticket class; and (4) the previous time period's prices (initialized to be 0 for the first time period). The choice of these features is based on our attempt to mimic what an airline would have access to in reality. Additionally, we incorporate scaling in order to allow for easier convergence in our chosen MARL optimization scheme (as shown in subsection 5.2). In our simulation, we differentiate between two types of profit: gross and net. Gross revenue and expenditure consists of data that airlines will typically have during the simulation period. This includes revenue generated from ticket sales and fixed costs associated with having tickets. Net revenue and expenditure, on the other hand, is concerned with data that an airline will only have access to after the time of flight departure. It concerns itself with actual realized revenue and expenditure at the end of the simulation period with things such as bump costs and no-show customers taken into account. In this sense, the airline seeks to maximize its net profit while however only having access to its gross profit. We more formally specify the difference between gross and net profit in the reward function specification of subsection 4.2.4.

Although there are potentially multiple customer agents at play, the observation vector $o_{i,t,c}$ adheres to the limited information mechanic such that a single customer only has access to information relevant to itself and cannot see the personal information of other customers. Upon reaching information state $s_{t,c}$ customer agent i's observation vector contains (1) the current timestep t scaled by the total timesteps before departure T; (2) their current spending capital; (3) a boolean indicator of their income bracket (lower, middle, or upper); (4) the cost of their ticket if they purchased one; (5) a boolean indicator of which ticket class they bought if they purchased a ticket; (6) the current prices set by the airline for each ticket class; and (7) the current number of available seats scaled by the total number of seats available per ticket class. For the information that requires a ticket to be bought, we default the values to 0 in the event that no ticket has been bought.

4.2.3 Action Space

Earlier in the beginning of 4.2, we defined an arbitrary airline action $\mathbf{a}_{t,a} \in \mathcal{A}(s_{t,a})$ to be $[p_{1,t}, \ldots, p_{j,t}, b_{1,t}, \ldots, b_{,t}j]^T$ where $p_{j,t}$ and $b_{j,t}$ are the price and booking limits respectively of ticket class j in time period t. Due to the need for dynamic prices and booking limits, we define $\mathcal{A}(s_{t,a})$ to be continuous. Additionally, we set $j = 3$ and bound the prices and booking limits to be between 0 and 1. With regards to the prices, the $[0, 1]$ bound is relatively straightforward. For the booking limit, however, we further define an interpretable transformation such that 0 denotes that no tickets may be sold and 1 allows for the airline to overbook, at most, two times

[2]For the sake of notational convenience, we use o and s interchangeably. In the event of any confusion, it suffices to understand that if an agent observes an information state s, they actually observe an agent-specific observation $o = o(s)$.

the ticket class limit (thus 4 for first class, 6 for second, and 10 for third). In the event that a non-integer number of tickets is allowed to be sold, we round to the nearest integer. Thus, for an explicit definition, an arbitrary $\mathbf{a}_{t,a} \in \mathcal{A}(s_{t,a})$ is defined to be

$$\mathbf{a}_{t,a} = [p_{1,t}, p_{2,t}, p_{3,t}, b_{1,t}, b_{2,t}, b_{3,t}]^T; \qquad p_{j,t}, b_{j,t} \in [0,1] \quad \text{for } j = 1, 2, 3.$$

Unlike the airline action space, the action space of customers is of a discrete nature. As mentioned earlier, $\mathbf{a}_{t,c} = [x_{1,t}, \ldots, x_{C,t}]^T$ where $x_{i,t}$ is the chosen action of customer agent i in time period t. In each time period, customers can choose to buy a ticket from any of the ticket classes or do nothing and thus $x_{i,t} \in \mathcal{X} := \{\text{do nothing}, \text{buy ticket class j}, \ldots\}$. To maintain a form of simplicity in the customers' total action space, we set that if a customer decides to purchase an arbitrary ticket A while already possessing another arbitrary ticket B, ticket B is cancelled. While this decision process is simple in principle, there are restrictions to when a customer is able to buy a ticket: if the number of tickets sold exceeds the booking limit or the customer does not have enough capital, the customer is not able to purchase the ticket.

4.2.4 Rewards

For all RL applications, the reward function plays a pivotal role in determining the resulting strategies. The rewards given after each EFG$_t$ is effectively the change in profit for the airline, and the change in utility for the customers.

Throughout the simulation upto but not including the day of departure, the airline reward is the change in its gross profit:

$$\Delta G_{\text{profit},t} = G_{\text{profit},t} - G_{\text{profit},t-1} = (G_{rev,t} - G_{exp,t}) - (G_{rev,t-1} - G_{exp,t-1}).$$

On the day of departure, however, the airline reward becomes the difference between net profit and gross profit: $N_{\text{profit},T} - G_{\text{proft},T-1}$. Gross expenditure ($G_{exp}$) is a fixed value where the investment cost associated with each ticket class is multiplied by the number of tickets available per ticket class. Upon implementation, we defined the investment costs to be 0.3 per first class ticket, 0.2 per second class, and 0.1 per third class—thus defining G_{exp} to be -1.7. Gross revenue (G_{rev}) simply keeps track of the sales: if a ticket is sold (cancelled) at a specific price, the revenue generated from that price is added (subtracted). In the event that a ticket is cancelled, the customers receive a cancellation cost of $\lambda = 0.2$ times the purchased ticket's price. This cost is added to the airline's G_{rev}. At the time of flight departure, bump costs are taken into account in the net revenue (N_{rev}) and expenditure (N_{exp}). Put more formally,

$$\begin{aligned} N_{rev} &= G_{rev} - (\text{bumped tickets' prices}) \\ N_{exp} &= G_{exp} + \beta(\text{bumped tickets' prices}), \end{aligned}$$

where β is the bump cost rate that we predefine to be 0.3. Using N_{rev} and N_{exp}, we define

$N_{\text{profit}} = N_{rev} - N_{exp}$. Thus the reward for the airline agent is defined to be

$$r_{t,a} = \begin{cases} G_{\text{profit},t} & \text{if } t = 0 \\ \Delta G_{\text{profit},t} & \text{if } t = 1, 2, \ldots, T-1 \\ N_{\text{profit},t} - G_{\text{profit},t-1} & \text{if } t = T. \end{cases}$$

While the airlines are mainly concerned with net profit, customers are tasked with maximizing their utility which is a function of their end capital and the utility gained from making it onto the flight. For the individuals that make it on the flight, the amount of utility gained is dependent on the ticket class. For first class, customers receive a utility of 1.0, second class gives 0.75, and third class gives 0.5. In the event that a customer has a bought a ticket and is bumped, they receive a full refund plus an extra monetary compensation equivalent to the bump cost rate β times the price they paid $k_{i,c}$. If we denote $q_{i,t}$ as the capital of customer i in period t, the reward function for customer agent i thus becomes

$$r_{i,t,c} = \begin{cases} q_{i,t} & \text{if } t = 0 \\ \Delta q_{i,t} & \text{if } t = 1, 2, \ldots, T-1 \\ \Delta q_{i,t} + (\text{ticket class utility}) & \text{if } t = T, \text{ ticket is bought, and not bumped} \\ \Delta q_{i,t} + \beta k_{i,c} & \text{if } t = T, \text{ ticket is bought, and bumped.} \end{cases}$$

5 Multi-Agent Policies

Earlier in Section 3, we described that the agents choose their actions based on a stochastic policy $\pi(s)$. In this section, we explore how these policies are defined and how we optimize them through the SAC algorithm in the multi-agent setting. A common highlight in the previous sections is the distinction between airline and customer agents having different observation vectors and action spaces. As the airline and customer agents act on different action spaces, their subsequent policy constructions are different. Thus we denote π_a and $\pi_{i,c}$ to be the respective policies of the airline agent and customer agent i.

5.1 Policy Definition

The choice of a stochastic policy allows for a natural approach towards random/deterministic exploration in the simulation environment. In our implementation, we make use of deep RL—meaning that we use parameterized policies characterized through neural networks. The choice of using neural networks is based on a few considerations. The first is that RL is based on the assumption that a pre-specified model learns the reward and transition functions without any prior knowledge by collecting experiences from the simulation environment. In the event that the reward functions and transitions are of a complicated nature, neural networks—provided they have a sufficient amount of parameters—adhere to the universal approximation theorem which states that neural networks are able to approximate any function (Csáji et al., 2001). The second consideration is based on the current trajectory of research surrounding advancements in RL. With the addition of neural networks, RL has seen state of the art results and many subsequent optimization schemes—including SAC—are built for neural networks. For brevity we omit the actual neural network architecture and move those details instead to Appendix B.

5.1.1 Airline Policy

Parameterizing the airline agent policy with θ_a, we define the stochastic policy of π_{θ_a} to be a neural network that outputs the defining characteristics of a diagonal Gaussian distribution. In other words, π_{θ_a} maps an arbitrary information state s to a mean μ_{θ_a} and log standard deviation $\log(\sigma_{\theta_a})$ vector. As a result, given an information state s, π_{θ_a} samples actions $\mathbf{a}'_{t,a}$ stochastically through

$$\mathbf{a}'_{t,a} \sim N(\mu_{\theta_a}(s), \sigma_{\theta_a}(s)I)$$

where N denotes the multivariate normal distribution and I is an identity matrix. If we instead want to sample actions deterministically, we simply set $\mathbf{a}'_{t,a} = \mu_{\theta_a}(s)$. To bind our sampled actions to the $[0, 1]$ bound, we apply a squashing transformation in the form of a sigmoidal function such that

$$\mathbf{a}_{t,a} = \Sigma(\mathbf{a}'_{t,a}) \tag{9}$$

where $\Sigma(x) = 1/(1 + \exp\{-x\})$. With regards to actual implementation, three ticket classes denote an action size of six (three prices, three booking limits) for our airline and thus $\pi_{t,a}$ receives the airline information state $s_{t,a}$ and outputs $\mu_{\theta_a}(s_{t,a})$ and $\log(\sigma_{\theta_a})(s_{t,a})$ of size six. From these vectors, the airline action $\mathbf{a}_{t,a} = [p_{1,t}, p_{2,t}, p_{3,t}, b_{1,t}, b_{2,t}, b_{3,t}]^T$ is sampled accordingly. Under this perspective, we note that the $\pi_{t,a}$ inherently outputs a joint action rather than a single action.

5.1.2 Customer Policy

On the other hand, we parameterize the customer agents' policies with $\theta_{i,c}$ such that $\pi_{\theta_{i,c}}$ is a neural network that outputs the defining features of a multinomial distribution. Put more formally, π_{θ_c} maps an arbitrary information state s to the probabilities of taking one of the discrete actions in $\mathcal{X}$ (previously defined in 4.2.3) forming a multinomial distribution. For our implementation, we additionally set that each of the customers follow the same policy such that $\theta_c = \theta_{1,c} = \ldots = \theta_{C,c}$. While this choice limits the degree to which customer strategies differ from one another, it ultimately eases the computation burden and forces the customer policy to generalize across all randomization set by the chance agent. From this multinomial distribution, the action of customer i, $x_{i,t}$, can be stochastically sampled through the multinomial distribution sampling scheme and deterministically through taking the element in $\mathcal{X}$ with the highest probability. Left alone, customer agents could choose any action in $\mathcal{X}$. In order to restrict actions possible under certain conditions, we additionally transform the outputted multinomial distribution such that the actions not allowed—if they exist—are set to have a probability of zero. As a result of this construction, the customer information states $s_{t,c}$ will be mapped by π_{θ_c} to C different multinomial distributions—one for each customer. From these C distributions, the joint action $\mathbf{a}_{t,c} = [x_{1,t}, \ldots, x_{C,t}]^T$ can be found.

5.2 Policy Optimization

As mentioned before, to find the optimal airline and customer policies, $\pi^*_{\theta_a}$ and $\pi^*_{\theta_c}$, we make use of the SAC algorithm explained in subsection 3.4. However, due to our hybrid simulation environment, we outline several optimization changes from the original algorithm.

The first distinction is that the objective function of the SAC algorithm from Equation 5 must be altered to accommodate both the airline and customer policies as well as the new hybrid environment formulation. Let us define $\pi = (\pi_{\theta_a}, \pi_{1,\theta_c}, \ldots, \pi_{C,\theta_c})$ to be the joint policy of both airline and customer agents. Then Equation 5 becomes agent i specific:

$$\pi^*_{\theta_{i,k}} = \underset{\theta_{i,k}}{\operatorname{argmax}}\, \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t R_{i,t}(\pi) + \alpha \mathcal{E}[\pi_{\theta_k}(\cdot|s_{t,k})]\Big|^{\mathbf{a}_{t,a}\sim\pi_{\theta_a}(\cdot|s_{t,a})}_{\mathbf{a}_{t,c}\sim\pi_{\theta_c}(\cdot|s_{t,c})}, s_0 \sim \pi_{chance}\Big] \quad k \in \{a, c\}, \quad (10)$$

where $R_{i,t}$ is the expected reward of EFG$_t$ (defined in Equation 4). From Equation 10, an extra degree of complication is added in that agent i's expected reward is dependent on the sequential joint actions of all agents. This necessitates an additional prediction of the other agents' actions.

Apart from the explicit change in equation formulation, the most important distinction is that a multi-agent setting introduces multi-agent specific problems such as scalability and non-stationarity (K. Zhang, Yang, and Başar, 2019). At its essence, scalability deals with the combinatorial nature of MARL (Hernandez-Leal, Kartal, and Taylor, 2019) such that the number of actions each agent has to consider grows exponentially with the number of agents. This proves detrimental to post-optimization analysis on convergence. Non-stationarity, on the other hand, is a result of multiple agents jointly optimizing at the same time. In order for algorithms such as SAC to converge in a respectable time, one of the properties that must hold is memoryless Markovian[3]. Otherwise known as stationarity, this property encompasses a time invariance associated with the simulation space such that an arbitrary EFG$_t$ encompasses the information of all previous EFGs. If each agent independently optimizes their own policies, stationarity is violated and convergence is not guaranteed (as explored in Tan, 1993 and Claus and Boutilier, 1998). Of these two issues, non-stationarity is the most problematic as scalability is not yet an issue for our implementation. As a result, we address a heuristic to circumvent the non-stationarity in the SAC optimization. To explain this heuristic, we split the optimization such that we consider the airline and customer agents separately. Otherwise, we provide the complete optimization pseudocode in Algorithm 2.

5.2.1 Airline Optimization

From an airline point of view, the airline Q-function approximator Q_{ϕ_a} must accurately predict the total expected entropy reward of the airline agent

$$\mathbb{E}\Big[\sum_{t\geq 0} \gamma^t R_{t,a}(\pi) + \alpha_a \mathcal{E}[\pi_{\theta_a}(\cdot|s_{t,a})]\Big], \quad (11)$$

[3]This property states that each state and individual-specific reward should *only* depend on the previous state and chosen action.

Algorithm 2 Airline/Customer MASAC Optimization

```
Input: airline/customer policy parameters $\theta_a$,$\theta_c$, Q-function parameters $\phi_{1,a}$, $\phi_{2,a}$, $\phi_{i,1,c}$, $\phi_{i,2,c}$
for $i = 1, \dots, C$ customers, initial $\alpha_a$, $\alpha_c$;
1. Set Q-function target parameters $\phi_{j,a}^{targ} \leftarrow \phi_{j,a}$ and $\phi_{i,j,c}^{targ} \leftarrow \phi_{i,j,c}$    for $j = 1, 2$;
2. Set $\mathcal{D} \leftarrow \emptyset$;
3. repeat
4.   Collect tuple $u = (s_{t,a}, s_{t,c}, z_t, r_{t,a}(d_t), r_{1,t,c}(d_t), \dots, r_{C,t,c}(d_t), s_{t+1,a}, s_{t+1,c})$ by running
     $\pi_{\theta_a}$ and $\pi_{\theta_c}$ in environment (see Algorithm 1);
5.   Set $\mathcal{D} \leftarrow \mathcal{D} \cup \{u\}$
6.   if time to update do
7.     for $l = 1, \dots,$ number of updates do
8.       Sample minibatch of tuples from $\mathcal{D}$;
9.       Update $\phi_{1,a}$ and $\phi_{2,a}$ with one gradient descent step using Equation 12;
10.      Update $\phi_{1,c}$ and $\phi_{2,c}$ with one gradient descent step using Equation 16;
11.      Update $\theta_a$ with one gradient descent step using Equation 14;
12.      Update $\theta_c$ with one gradient descent step using Equation 17;
13.      Update $\alpha_a$ with one gradient descent step using Equation 15;
14.      Update $\alpha_c$ with one gradient descent step using Equation 18;
15.      Update $\phi_{j,k}^{targ} \leftarrow \rho\phi_{j,k}^{targ} + (1-\rho)\phi_{j,k}$    for $j = 1, 2$   $k \in \{a, c\}$;
16.    end for
17.  end if
18. until convergence
```

where $R_{t,a}$ is the airline-specific reward from Equation 4 of the EFG in time period t. As a rule of thumb, in order for Q_{ϕ_a} to accurately approximate Equation 11, it must have all information associated with the production of the actual reward $r_{t,a}(s_{t,a}, s_{t,c}, z_t)$. With this consideration, we determine that instead of Q_{ϕ_a} *only* having access to the airline's information states and actions, $Q_{\phi_a}(s_{t,a}, \mathbf{a}_{t,a})$, it should have access to the information states and actions of *both* the airline and customer agents: $Q_{\phi_a}(s_{t,a}, s_{t,c}, z_t)$—making Q_{ϕ_a} a centralized Q-function. For notational convenience in the upcoming equations, we rewrite the Q-function to be $Q_{\phi_a}(s_{t,a}, s_{t,c}, \mathbf{a}_{t,a}, \mathbf{a}_{t,c})$. As the centralized Q-function determines the policy updates in the SAC algorithm, we are additionally able to confirm stationarity in the optimization environment (K. Zhang, Yang, and Başar, 2019; Lowe et al., 2017).

From these changes, we can formally write the new SAC optimization objectives. Let us assume that we have a trajectory storage $\mathcal{D}$ that stores tuples $(s_{t,a}, s_{t,c}, z_t)$ and that we additionally breakdown z_t into its $\mathbf{a}_{t,a}$ and $\mathbf{a}_{t,c}$ parts such that $d_t := (s_{t,a}, s_{t,c}, \mathbf{a}_{t,a}, \mathbf{a}_{t,c})$. In order to solve Equation 10 for the airline, Equation 6 changes to become

$$\min_{\phi_{j,a}} \mathbb{E}_{d_t \sim \mathcal{D}} \Big[\frac{1}{2}(Q_{\phi_{j,a}}(d_t) - y(d_t))^2\Big] \qquad j = 1, 2 \tag{12}$$

with

$$y(d_t) = r_{t,a}(d_t) + \gamma \mathbb{E}_{\substack{s_{t+1,a} \sim \mathcal{I}(z_t) \\ \mathbf{a}_{t+1,a} \sim \pi_{\theta_a}(s_{t+1,a}) \\ s_{t+1,c} \sim \mathcal{I}(\mathbf{a}_{t+1,a}) \\ \mathbf{a}_{t+1,c} \sim \pi_{\theta_c}(s_{t+1,c})}} \Big[\min_{j=1,2} Q_{\phi_{j,a}}^{targ}(d_{t+1}) - \alpha_a^T \log \pi_{\theta_a}(\mathbf{a}_{t+1,a}|s_{t+1,a})\Big] \tag{13}$$

where α_a^T is a vector of temperature parameters instead of a single temperature parameter. The construction of Equation 13 is based on the sequential order in which airline-specific rewards are generated: (1) the airline's information state is received from the EFG in the previous time period, (2) the airline chooses an action based on their information state observations, (3) the customers' information state observations are generated through the action of the airline's action, (4) the customers choose their actions—in order for the optimization to provide suitable gradient estimates, this order of sampling must be followed. Subsequently, Equation 7 becomes

$$\min_{\theta_a} \mathbb{E}_{\substack{s_{t,a} \sim \mathcal{D} \\ \mathbf{a}_{t,a} \sim \pi_{\theta_a}(s_{t,a}) \\ s_{t,c} \sim \mathcal{I}(\mathbf{a}_{t,a}) \\ \mathbf{a}_{t,c} \sim \pi_{\theta_c}(s_{t,c})}} \Big[\alpha_a^T \log \pi_{\theta_a}(\mathbf{a}_{t,a}|s_{t,a}) - \min_{j=1,2} Q_{\phi_{j,a}}(s_{t,a}, s_{t,c}, \mathbf{a}_{t,a}, \mathbf{a}_{t,c})\Big] \tag{14}$$

and Equation 8 is rewritten as

$$\min_{\alpha_a} \mathbb{E}_{\substack{s_{t,a} \sim \mathcal{D} \\ \mathbf{a}_{t,a} \sim \pi_{\theta_a}(s_{t,a})}} \Big[- \alpha_a^T \log \pi_{\theta_a}(\mathbf{a}_{t,a}|s_{t,a}) + H_a\Big]. \tag{15}$$

5.2.2 Customer Optimization

Similar to the airline optimization, the customer optimization makes use of a centralized Q-function approximator. However, a key difference between the two is in the fact that the discrete customer policy π_{θ_c} characterizes a complete distribution—namely a multinomial. Christodoulou (2019) note that a policy (such as π_{θ_a}) that outputs a continuous density must require a monte-carlo estimate of the expectations. In contrast, the complete outputted distribution of π_{θ_c} allows for direct calculation of the expectations and thus they provide an updated SAC algorithm for discrete actions. In the event that there a small number of agents and actions, direct derivation of the expectation remains feasible. However, this direct derivation quickly becomes infeasible as the number of agents or actions increase. For example, as we consider twenty customer agents with four actions each, direct calculation of the expectation requires a computation of approximately one trillion action combinations—thus infeasible. To this end, we slightly alter the discrete SAC algorithm of Christodoulou (2019).

In order to optimize Equation 10, let us assume that $\mathcal{C}$ denotes the set of customer agents. Then the Q-function approximator of the ith customer $Q_{\phi_{i,c}}$ receives (1) the information states $s_{t,a}$ and $s_{t,c} = [o_{1,t,c}(s_{t,c}), \ldots, o_{C,t,c}(s_{t,c})]^T$ of the airline agent and all customers agents respectively (thus relaxing the assumption that customer agents can not see each other's information states); (2) the actions $\mathbf{a}_{t,a}$ and $\mathbf{a}_{-i,t,c} = [x_{1,t}, \ldots, x_{i-1,t}, x_{i+1,t}, \ldots, x_{C,t}]^T$ of the airline agent and all $\mathcal{C} \setminus i$ customers respectively. Using these inputs, $Q_{\phi_{i,c}}(s_{t,a}, s_{t,c}, \mathbf{a}_{t,a}, \mathbf{a}_{-i,t,c})$ outputs a state-action value for *all* possible actions (instead of for a single action) for customer $i \in \mathcal{C}$. Thus, a local expectation for customer i can be calculated using the outputted distribution $\pi_{\theta_c}(o_{i,t,c})$. If we assume that tuples of $(s_{t,a}, s_{t,c}, z_t)$ are stored in a trajectory storage $\mathcal{D}$ where $d_t := (s_{t,a}, s_{t,c}, \mathbf{a}_{t,a}, \mathbf{a}_{t,c})$ and $d_{-i,t} := (s_{t,a}, s_{t,c}, \mathbf{a}_{t,a}, \mathbf{a}_{-i,t,c})$, the optimization of the ith customer Q-function thus becomes

$$\min_{\phi_{i,j,c}} \mathbb{E}_{\substack{d_t\sim\mathcal{D} \\ i\sim\mathcal{C}}}\Big[\frac{1}{2}(Q_{\phi_{i,j,c}}(d_{-i,t}) - y(d_t, d_{-i,t}))^2\Big] \qquad j = 1,2 \tag{16}$$

where

$$y(d_t, d_{-i,t}) = r_{i,t,c}(d_t) + \gamma \mathbb{E}_{\substack{s_{t+1,a}\sim\mathcal{I}(z_t) \\ \mathbf{a}_{t+1,a}\sim\pi_{\theta_a}(s_{t+1,a}) \\ s_{t+1,c}\sim\mathcal{I}(\mathbf{a}_{t+1,a}) \\ \mathbf{a}_{t+1,c}\sim\pi_{\theta_c}(s_{t+1,c})}}\Big[\pi_{\theta_c}(o_{i,t+1,c})^T\Big(\min_{j=1,2} Q^{targ}_{\phi_{i,j,c}}(d_{-i,t+1}) - \alpha_c \log \pi_{\theta_c}(\cdot|o_{i,t+1,c})\Big)\Big].$$

From here we optimize θ_c and α_c by averaging the results of the following objective functions across each customer i:

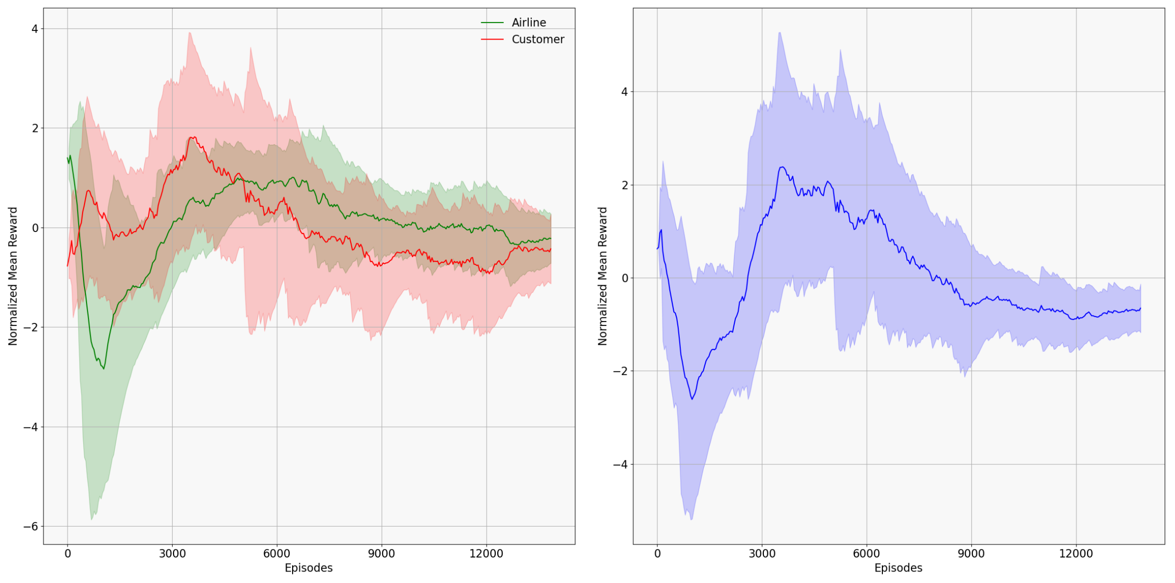
$$\min_{\theta_c} \mathbb{E}_{\substack{s_{t,a}\sim\mathcal{D} \\ \mathbf{a}_{t,a}\sim\pi_{\theta_a}(s_{t,a}) \\ s_{t,c}\sim\mathcal{I}(\mathbf{a}_{t,a}) \\ \mathbf{a}_{t,c}\sim\pi_{\theta_c}(s_{t,c}) \\ i\sim\mathcal{C}}}\Big[\pi_{\theta_c}(o_{i,t,c})^T\Big(\alpha_c \log \pi_{\theta_c}(\cdot|o_{i,t,c}) - \min_{j=1,2} Q_{\phi_{i,j,c}}(d_{-i,t})\Big)\Big] \tag{17}$$

$$\min_{\alpha_c} \mathbb{E}_{\substack{s_{t,c}\sim\mathcal{D} \\ i\sim\mathcal{C}}}\Big[\pi_{\theta_c}(o_{i,t,c})^T\Big(\alpha_c(\log \pi_{\theta_c}(\cdot|o_{i,t,c}) + H_c)\Big)\Big]. \tag{18}$$

6 Results

To train the airline and customer agents in the simulation space, we implement Algorithm 2 for a total of 1.4 million environment steps (amounting to 14000 episodes). To visualize the optimization process, we include Figure 5 which shows the normalized mean rewards during the course of the training for both airline and customer agents. In order to better visualize the progression, we consider only the rewards obtained from simulations where there are 10 customers. For the optimization itself, we include our chosen hyperparameters in Table 1.

In the beginning phases of the training, we see that the airline and customer rewards portray a negative correlation effect: when the airline rewards increase, the customer rewards decrease. As we consider a competitive environment, this relationship is to be expected. However, as training continues, we see that this relationship does not remain consistent. In fact, looking at Figure 5a, we see that there are even instances where the airline's reward becomes positively correlated with customer's reward. Because of this, we hypothesize that while the environment space is inherently competitive, there is a still a level of cooperation involved between the airline and customers to achieve mutual improvement. From a convergence perspective, the analysis becomes more nuanced. For any multi-agent simulation setting, Shoham, Powers, and Grenager (2003) note that it is difficult to determine the actual optimization goals adding that, in general, Nash equilibrium (NE) is the main goal for multi-agent optimization. However, as we periodically mention throughout the thesis, reality is messy and the theoretical assumptions of NE can be difficult to meet. Additionally, as our goal for this thesis was to assert whether or not MARL can adequately approximate reality, we abstain from an analysis into the NE and instead focus on the intelligence of the optimized agents and whether or not they possess



(a) Separate **(b)** Combined

Figure 5: Reward Optimization
This figure consists of plots of the normalized 10 customer episode reward progression throughout the course of training, smoothed with an exponential moving average. In particular, 5a plots the normalized reward for the airline and customer agents separately whereas 5b plots the sum of the two normalized rewards.

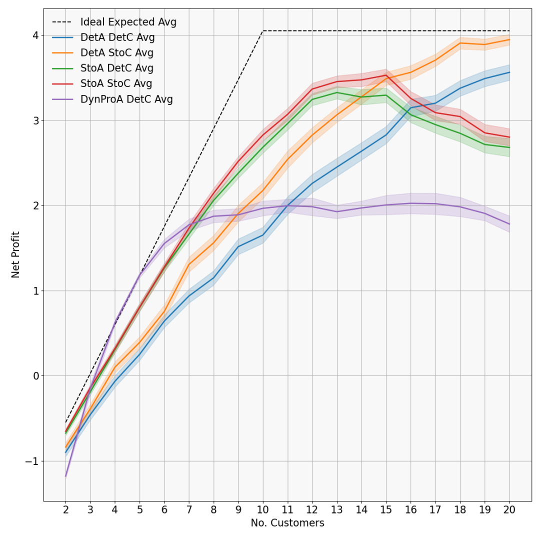
human-like behavior.

6.1 Agent Intelligence

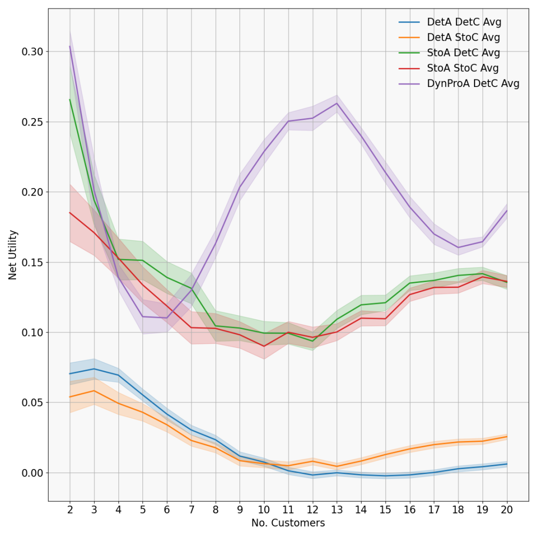
As our research is rooted in revenue management, we begin our intelligence investigation into whether or not the airline agent is successfully able to perform respectably in the sense of generating revenue and profit. After this, we then analyze the strategies taken by the customers to affirm that the airline is simulated against suitably intelligent agents. Throughout the analysis, we will continuously compare the results to what we may intuitively expect from real life.

6.1.1 Airline Performance

To visualize the overall performance of the airline agent, we provide Figure 6 which displays the average net profit and utility (as well as their 95% confidence intervals) obtained by the airline and customer agents respectively for a varying number of customers and action sampling schemes. In subsection 5.1, we explained that actions can be sampled deterministically (Det) or stochastically (Sto). For a complete analysis, we thus perform 300 simulations with every possible action sampling combination. Additionally, for comparison purposes, subfigure 6a additionally includes: (1) a theoretical *ideal expected avg* dotted line that displays the expected profit the airline could earn in the idealized situation where the customers buy every available ticket with their expected wealth; (2) the results of an approximate dynamic programming approach that determines the airline's strategy against the optimized MASAC customers. If interested, the details behind both comparison methods can be found in Appendix A. We see that the general trend is an increase in net profit and decrease in net utility as the number of customers increase. This naturally makes sense considering the laws of supply and demand. While acting stochastically initially provides a higher performance for the airline, we see that around 12-14 customers, acting deterministically becomes a more viable strategy. This is to be expected as we understand that past 10 customers, the airline begins to deal with overbooking issues and their associated bumping risks. However, when acting deterministically, we see that the airline is able to continuously increase its profit even when the demand for outweighs the supply suggesting that it has developed a strategy that successfully incorporates overbooking while minimizing the bumping risks.



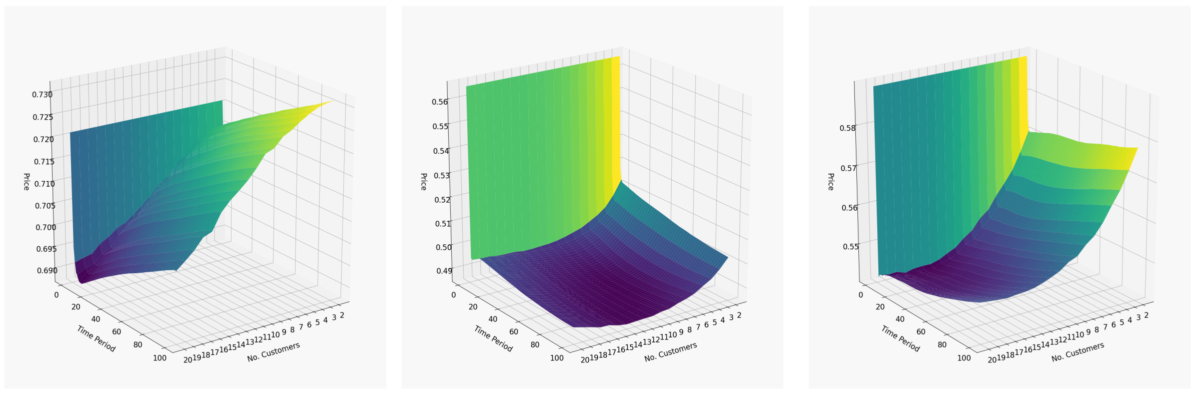
(a) Airline Avg Net Profit



(b) Customer Avg Net Utility

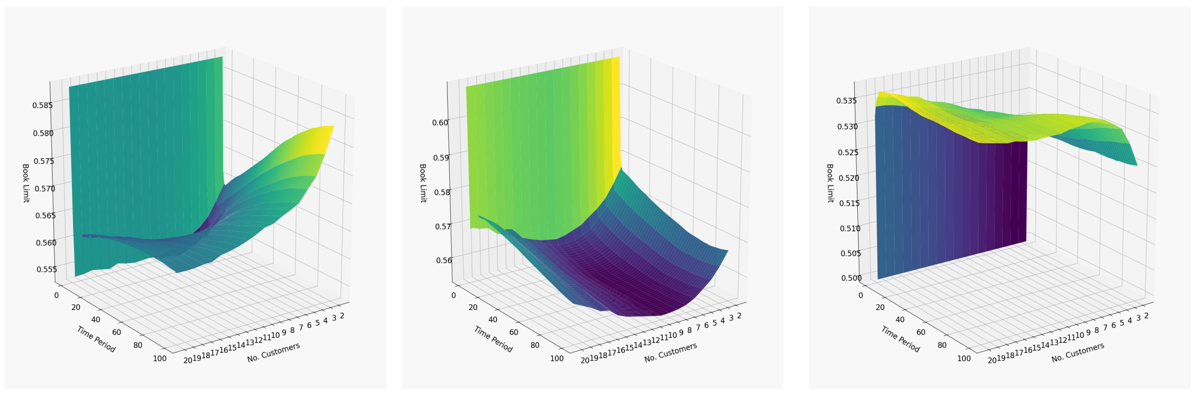
Figure 6: Net Profit/Utility
This figure shows the average net profit and utility (and their 95% confidence intervals) obtained by the airline (A) and customer (C) agents respectively for 2 to 20 customers. We additionally include the results for different combinations of the action sampling schemes (deterministically (Det) and stochastically (Sto)) as well as the results of an approximate dynamic programming (DynPro) approach that optimizes the airline's strategy against the MASAC customers. As we deal with net profit in 6a, the initial investment cost of -1.7 is automatically included. The averages and confidence intervals are obtained through 300 simulation runs per number of customers. For the airline net profit, we additionally include an *ideal expected avg* line that plots the expected net profit in the idealized situation where the airline can sell every ticket at a price equal to the expected amount of capital per customer. This line does not consider the potential profit gain from overbooking tickets.

Turning to Figures 7 and 8, we display 3D plots of the prices and booking limits respectively set by the airline over time and varying customer sizes. At an initial glance, we see a common pattern where the transition from time period 0 to 1 is met with an abrupt change in action value. Additionally, the resulting values after these abrupt changes differ depending on the number of customers. What this essentially portrays is that the airline agent adopts a sort of sampling mechanism that provides an indication to the airline of the number of customers present and thus, the expected demand. As the number of customers acting in the simulation is randomly determined and the airline has no knowledge of this exact number, the airline agent devises a method to discover this information by itself. Inherently, this displays the airline agent's ability to approximate the customers' strategies. Besides these points, it is interesting to note that while first class tickets are naturally the most expensive, the airline agent actually sets second class tickets to be the cheapest with third class tickets in the middle. This is in contrast to how one would naturally expect higher quality to correlate with higher prices. However, if we consider Figures 11 and 12, we do see that the demand for second class tickets is quite high. Thus the lower second class tickets may be a strategic choice on the airline's side.



(a) 1st Class **(b)** 2nd Class **(c)** 3rd Class

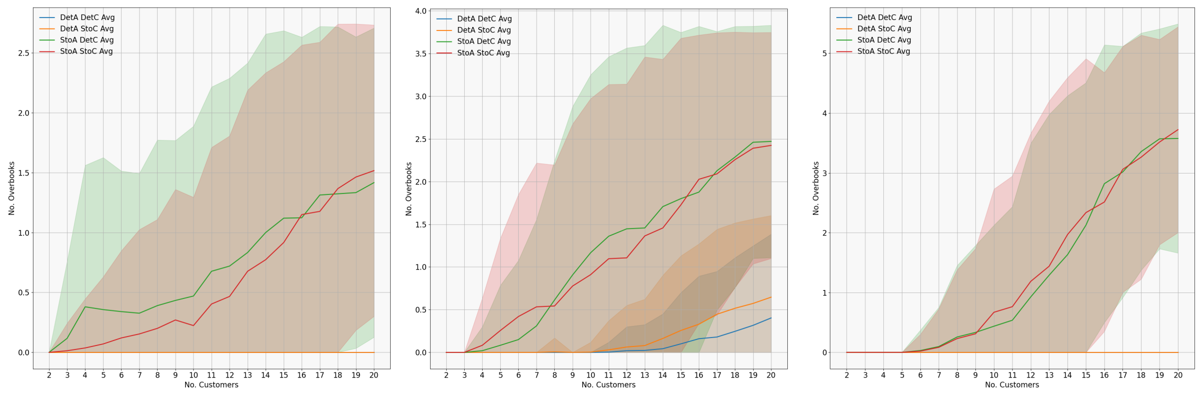
Figure 7: Average Price per Ticket Class
This figure consists of surface plots of the average price for each ticket class over time and number of customers. The averages are calculated based on 300 simulation runs per customer size where the airline and customer agents' actions were deterministically sampled.



(a) 1st Class **(b)** 2nd Class **(c)** 3rd Class

Figure 8: Average Booking Limit per Ticket Class
This figure consists of surface plots of the average booking limit for each ticket class over time and number of customers. The averages are calculated based on 300 simulation runs per customer size where the airline and customer agents' actions were deterministically sampled.

If we look at the strategy adopted for first class tickets, we see that, in general, the airline agent sets the adjusted price to be lower as the number of customers in the simulation increases. From here, the price increases over time. The booking limit for first class tickets, on the other hand, follows a different format. After the initial sampling period, the booking limit is set to be U-shaped such that around 9-12 customers, the booking limit is at its lowest point. Interestingly enough, this coincides with the simulation scenario where demand approximately meets supply. Naturally the airline agent is not exact as it is only able to estimate the number of customers instead of knowing directly. Moreover, similar to the price, the airline agent generally increases the booking limit over time. In contrast, the second class ticket strategy actually lowers its price and booking limit over time while providing an even stronger emphasis on the U-shape approach. In the event of price elastic customers, the second class ticket would thus actually become more attractive over time as opposed to the first/third class tickets. As such, if the airline is able to attract the remaining customers to the second class tickets over time, it is able to increase the likelihood of no-show customers to buy second class tickets mitigating the risk involved with overbooking second class tickets. In other words, the higher concentrated demand reduces the risks involved with overbooking. If we additionally consider Figure 9, we see that when acting deterministically, the airline also only overbooks the second class tickets. The strategy for third class tickets looks similar to that of the first class ticket where it decreases the initial price after the sampling period. However, its trend over time is more or less stable. Its booking limit is noticeably different from all other strategies. While all other ticket classes have their booking limits and prices lowered after the sampling period, the third class booking limit increases instead with the number of customers. This points to the observation that the third class ticket may have the highest demand of the three ticket classes. If the airline sets a high booking limit for the sampling of customers, the large demand may cause it to overbook to a point where the net profit suffers. Thus, it is more conservative initially. This hypothesis is further corroborated by Figures 11 and 12 which show the average demand for each type of customer over time in comparison to the price.

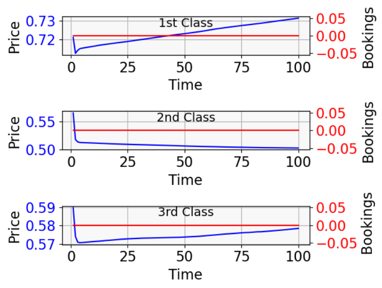


(a) 1st Class **(b)** 2nd Class **(c)** 3rd Class

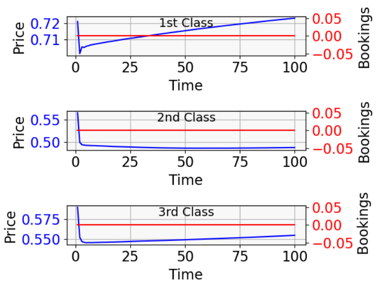
Figure 9: Average No. Over-bookings per Ticket Class
This figure consists of plots of the average number of tickets overbooked per ticket class for each MASAC sampling scheme. DetA and StoC stand for when the airline samples actions deterministically and customers sample actions stochastically respectively. Besides showing the average, the plots additionally show the standard deviation of the number of bookings incorporating approximately 96% of the observations. The averages and standard deviations are calculated based on 300 simulation runs per customer size.

From a general perspective, an interesting strategy that the airline agent has settled on is to not sell any tickets to lower class customers. As mentioned before, lower class customers are randomly initialized to have a wealth between 0.25 and 0.5. Looking at the prices, the ticket prices are simply too high for lower class customers to purchase. This can, of course, be seen in Figure 10. Based on this, we can theorize a sort of collaboration between the airline and customers. By not selling tickets to lower class customers (who have been set to be 30% of the total customer population), the airline significantly reduces the overall competition for tickets for both the middle and upper class customers. Naturally, to compensate for this decision, the airline is able to sell the tickets at higher prices. Abstaining from all ethical considerations and any government-based intervention, this catering type of strategy is not impossible in reality when considering a monopolistic environment such as the one in our simulation. It is likely that if the simulation consisted of another airline, catering strategies would be less prevalent.

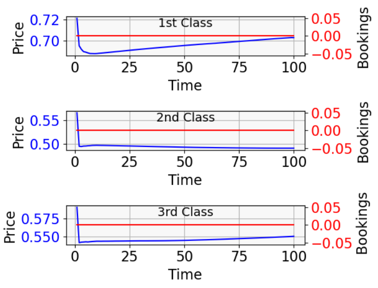
6.1.2 Human-like Customers



(a) 2 Customers



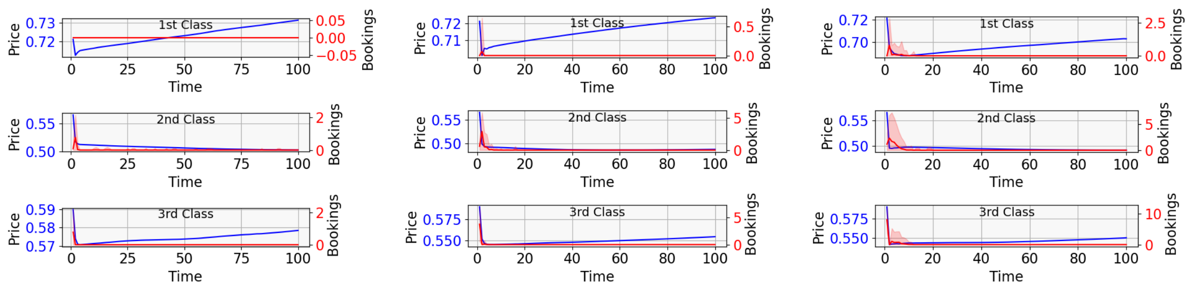
(b) 10 Customers



(c) 20 Customers

Figure 10: Average No. Bookings for Lower Class Customer
This figure consists of plots of the average price compared to the average number of bookings for customers who are initialized to be lower class. Besides showing the average, the plots additionally show the standard deviation of the number of bookings incorporating approximately 96% of the observations.

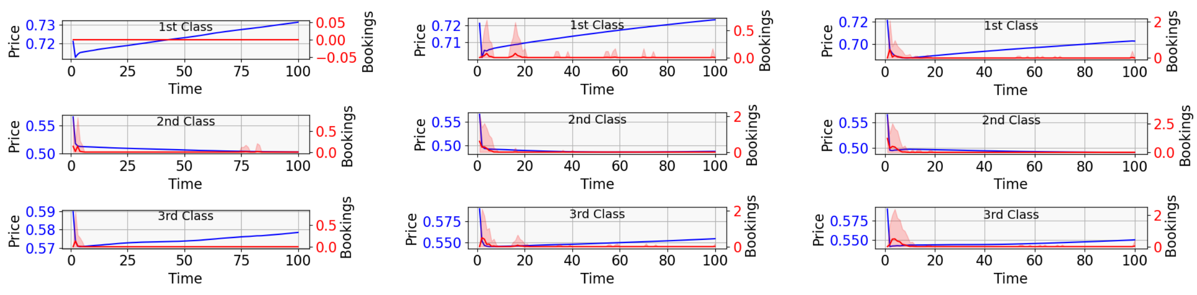
Due to the joint optimization nature of Algorithm 2, the evolution of the airline agent is intuitively limited by the customer agents' evolution and vice versa. As a result, based on the resulting strategies of the airline agent, we naturally expect the customers to exhibit a modicum of intelligence. To gain a better understanding of the customer intelligence, we consider Figures 10, 11, and 12 where we analyze each customer type's behavior in settings where supply exceeds demand (2 customers), supply equals demand (10 customers), and demand exceeds supply (20 customers). Unfortunately, due to the airline's choice of not selling tickets to lower class customers, we omit a discussion of lower class customers and instead consider only middle and upper class customers.



(a) 2 Customers **(b)** 10 Customers **(c)** 20 Customers

Figure 11: Average No. Bookings for Middle Class Customer
This figure consists of plots of the average price compared to the average number of bookings for customers who are initialized to be middle class. Besides showing the average, the plots additionally show the standard deviation of the number of bookings incorporating approximately 96% of the observations.

Initialized with a wealth between 0.5 and 0.75 as well as being the most prevalent customer class, the middle class customers are best described as being the most involved in the simulation environment. Similar to the airline, each individual customer does not have any knowledge of the number of active customers and thus the customers adopt a sampling mechanism as well. In this sampling mechanism, a large portion of the customers seek to purchase a ticket in the initial period. Depending on the number of successful ticket purchases, the customers are able to determine the approximate competition level. With a reference to Figure 11, we see that when supply exceeds demand, the middle class customers are met with freedom that comes from a lack of ticket competition. As a result, we see that they exhibit a form of price elasticity—having a tendency to purchase the second class ticket over time as its price decreases while avoiding the first/third class tickets whose prices increase over time. On the other hand, when the demand increases to be greater than or equal to supply, the competition for tickets increases accordingly. At this point, the middle class customers all purchase tickets as soon as they can after the sampling period mitigating the risk of not getting a ticket. The airline naturally increases the competition with the earlier mentioned demand concentration strategy. Considering the wealth associated with middle class customers, first class tickets quickly become unobtainable over time with third class tickets following suit. As such, the middle class customers feel the brunt of the airline's demand concentration strategy as they congregate to demand for the second class tickets. In the event that an airline adopted a demand concentration strategy to mitigate overbooking risk, it is not unnatural to expect customers in real life to act as the customer agents. This ultimately shows that middle class customers possess an ability to consider the risks associated with their financial settings and an inherent knowledge of how the airline acts.



(a) 2 Customers **(b)** 10 Customers **(c)** 20 Customers

Figure 12: Average No. Bookings for Upper Class Customer
This figure consists of plots of the average price compared to the average number of bookings for customers who are initialized to be upper class. Besides showing the average, the plots additionally show the standard deviation of the number of bookings incorporating approximately 96% of the observations.

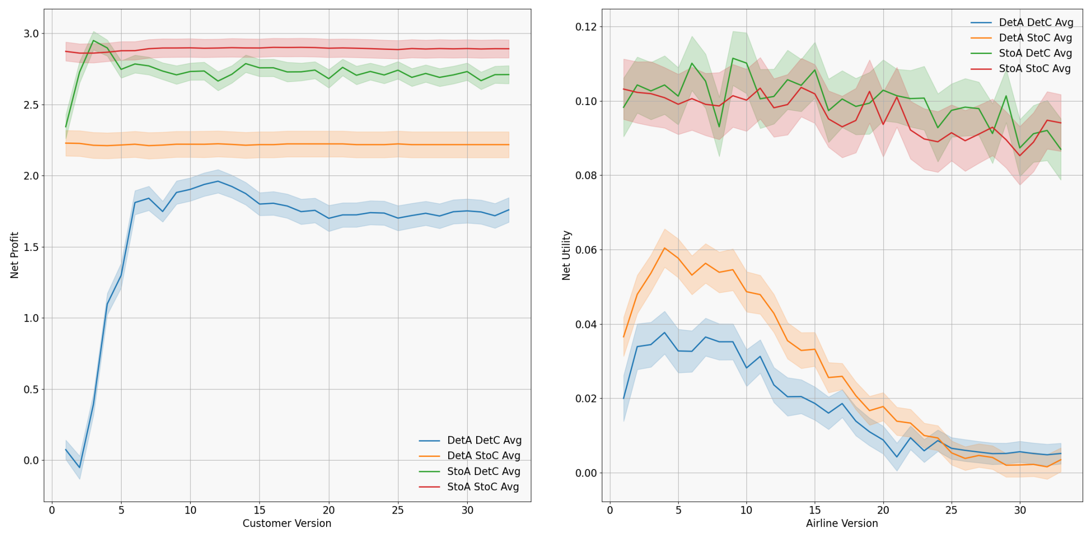
Unlike the middle class customers, upper class customers have more freedom in their choices as they have more starting capital. In fact, with a starting capital on the interval $(0.75, 1.0]$, the upper class customers wealth exceeds the maximum price the airline sets for any ticket class. The expanded freedom is most evident in Figure 12a where supply vastly exceeds demand. In this scenario, we see that the upper class customers purchase tickets at approximately their lowest prices (after the initial sampling period). Because of their wealth, the upper class customers are able to purchase a ticket at any point in time and thus they are able to wait as long as they need to get the best price. When there are 10 customers, the competition exhibited by middle class customers is still not present for upper class customers as the upper class customers possess much more capital. We can see in Figure 12b that when the airline adopts its demand concentration strategy, forcing middle class customers to the second class tickets, the upper class customers move their demand to the first and third class tickets. Thus, we also see that the airline caters to upper class customers with the demand concentration strategy. As supply is directly equal to demand with 10 customers, the upper class customers are guaranteed to receive a ticket eventually. At 20 customers where demand far exceeds supply, we see the upper class customers finally feeling the effects of the ticket competition as their guarantee of getting a ticket is no longer secure. As a result, we see in Figure 12c that the upper class customers purchase tickets as soon as possible.

Based on these considerations, we can see that the customers possess a certain degree of intelligence—especially in their ability to adapt to varying environment settings. Although we omit an analysis into the lower class customers, the behavior of the middle and upper class customers can be argued to be representative of what one might expect from the respective customer classes in reality.

6.2 Generalization

Despite the interesting strategies taken by both the airline and customer agents, we do observe that the agents' strategies are very dependent on one another. In fact, Lowe et al. (2017) note that in multi-agent optimization, there is a risk for agents to overfit on each other's behavior. Thus, the airline's resulting policy may fail when the customers act differently and vice versa. To analyze this, we simulate our optimized agents against older versions of themselves. Throughout the course of our training, we saved checkpoints of the agents every 400 episodes amounting to approximately 35 different agent versions. As such, we simulate the optimized airline agent against different customer versions and the optimized customer agents against different airline versions. The results of these simulations can be seen in Figure 13.

In Figure 13a, we see that the performance of the optimized airline agent remains relatively constant through the different customer versions. The only exception is for when the customer samples actions deterministically. In this scenario, the airline's performance can be considered subprime in the earlier customer versions. As this unstable performance occurs when the customer agents are in the early stages of training, we can argue that the airline performs poorly when the customer agents have not yet achieved an appropriate level of intelligence. On the other hand, when we compare the optimized customer agent to the varying airline agents in Figure 13b, we clearly see a decrease in performance as the airline agents become more intelligent. This naturally makes sense considering that the airline itself has a considerable amount of power in our monopolistic simulation space. However, it is most striking to see that the optimized customer agent has developed a policy that is able to perform relatively well regardless of the airline version, performing well when the airline gives it the opportunity to do so. Overall, there is no clear indication that the optimized agents have overfit to one another. If this was case, we'd see a clear degradation in performance the more the strategies diverge from the final optimized policies.



(a) Final Airline Avg Net Profit **(b)** Final Customer Avg Net Utility

Figure 13: Final vs Old Agents
This figure shows the average net profit and utility (and their 95% confidence intervals) obtained by the optimized airline (A) and customer (C) agents respectively when pitted against older versions of the opposing agent in a 10 customer simulation. Thus in Figure 13a, the optimized airline agent plays against older versions of the customers and in Figure 13b, the optimized customer agent plays against older versions of the airline. We additionally include the results for different combinations of the action sampling schemes (deterministically (Det) and stochastically (Sto)). The averages and confidence intervals are obtained through 300 simulation runs per agent version.

7 Discussion

In this thesis, we presented a multi-agent reinforcement learning simulator for the single-leg airline revenue management problem. Along with providing optimization details, we optimized two types of agents (airline and customer) and found that the final agents were capable of adapting to changes in their environment while still performing reasonably well. When analyzing the behavior of the agents, we found noticeable similarities to what one would expect from a reality where the simulation-specific circumstances hold—namely sampling mechanisms, market manipulation to mitigate risk, and customer class catering.

Although the created environment is limited by its varying assumptions and may not hold exactly in reality, the results presented can be used as a proof of concept for the potential of MARL. In other words, the more realistic a simulated environment, the more applicable the optimized agents become. This particular insight can be beneficial to highly complex applications of human-based interaction. For companies such as airlines, being able to simulate expected customer response allows for the creation of artificial data that help with determining strategies in hypothetical situations.

Naturally, with the potential benefits, there are also problems involved with MARL and the construction of simulation environments. Firstly, the optimized agents are limited by the environment they optimize themselves in. If a created environment does not accurately approximate reality, the optimized agents will be less applicable. This includes the actions that each agent has access to as well as the construction of the agent-specific observations and rewards. In particular, the reward function is by far the most important issue as it will determine the resulting strategies. With our simulation, our airline agent had a reward function that focused entirely on maximizing its end profit and because of this, the airline developed ethically-questionable strategies by only catering to its richer customers. Thus, one can argue that future reward functions must include ethical considerations as well. Additionally, the degree of adaptability that the agents demonstrate is also dependent on the environment space. For example, in our airline environment, the stochastic initialization allowed for the agents to develop their own adaptability. Of course, reality is complicated and the more exact an environment becomes, the larger the optimization problem becomes as well. Secondly, RL and MARL optimization is an ongoing research topic and is thus far from being optimal. There is no singular way to solve RL problems and thus issues such as algorithm/hyperparameter selection become application specific—especially for MARL, there is often no guarantee of convergence.

However, we do additionally consider potential workarounds for the aforementioned problems. For limiting the problem size, environment construction must strike a balance between specificity and generality. Just as in a controlled experiment, the environment can be specific on certain important characteristics while remaining general on less important details. While this may still result in a large optimization problem, the overall size will still have been significantly reduced. With regards to reward function construction, the problem becomes application specific and we can only suggest to consider what the goals are as well as analyzing what drives individuals to act a certain way. For companies, reward functions can perhaps be built with data collected on customer-behavior and/or with a pre-specified goal in mind. Additionally, while there are limitations associated with the present level of RL/MARL research, the current methods thus far have been seen to perform well—each with their own recommendation on the hyperparameter settings. As such, beginning a MARL application is not akin to fumbling in the dark.

8 Acknowledgements

Thank you to Professor Birbil for all his valuable insight and suggestions throughout the course of this entire thesis process.

References

Aydın, Nurşen et al. (2013). “Single-leg airline revenue management with overbooking”. In: *Transportation Science* 47.4, pp. 560–583.

Baker, Bowen et al. (2019). “Emergent tool use from multi-agent autocurricula”. In: *arXiv preprint arXiv:1909.07528*.

Bazzan, Ana LC (2009). “Opportunities for multiagent systems and multiagent reinforcement learning in traffic control”. In: *Autonomous Agents and Multi-Agent Systems* 18.3, p. 342.

Bellman, Richard (1966). “Dynamic programming”. In: *Science* 153.3731, pp. 34–37.

Bellman, Richard E and Dreyfus, Stuart E (2015). *Applied dynamic programming*. Princeton university press.

Bertsimas, Dimitris and De Boer, Sanne (2005). “Simulation-based booking limits for airline revenue management”. In: *Operations Research* 53.1, pp. 90–106.

Birbil, Ş İlker et al. (2014). “A network airline revenue management framework based on decomposition by origins and destinations”. In: *Transportation Science* 48.3, pp. 313–333.

Christodoulou, Petros (2019). “Soft actor-critic for discrete action settings”. In: *arXiv preprint arXiv:1910.07207*.

Claus, Caroline and Boutilier, Craig (1998). “The dynamics of reinforcement learning in cooperative multiagent systems”. In: *AAAI/IAAI* 1998.746-752, p. 2.

Csáji, Balázs Csanád et al. (2001). “Approximation with artificial neural networks”. In: *Faculty of Sciences, Etvs Lornd University, Hungary* 24.48, p. 7.

Fujimoto, Scott, Van Hoof, Herke, and Meger, David (2018). “Addressing function approximation error in actor-critic methods”. In: *arXiv preprint arXiv:1802.09477*.

Gosavi, Abhijit, Ozkaya, Emrah, and Kahraman, Aykut F (2007). “Simulation optimization for revenue management of airlines with cancellations and overbooking”. In: *Or Spectrum* 29.1, pp. 21–38.

Gosavii, Abhuit, Bandla, Naveen, and Das, Tapas K (2002). “A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking”. In: *IIE transactions* 34.9, pp. 729–742.

Gupta, Shubham and Dukkipati, Ambedkar (2019). “Probabilistic View of Multi-agent Reinforcement Learning: A Unified Approach”. In:

Haarnoja, Tuomas et al. (2018). “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *arXiv preprint arXiv:1801.01290*.

Hernandez-Leal, Pablo, Kartal, Bilal, and Taylor, Matthew E (2019). “A survey and critique of multiagent deep reinforcement learning”. In: *Autonomous Agents and Multi-Agent Systems* 33.6, pp. 750–797.

Kaelbling, Leslie Pack, Littman, Michael L, and Moore, Andrew W (1996). “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4, pp. 237–285.

Kingma, Diederik P and Ba, Jimmy (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Lin, Kaixiang et al. (2018). “Efficient large-scale fleet management via multi-agent deep reinforcement learning”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1774–1783.

Littman, Michael L (1994). “Markov games as a framework for multi-agent reinforcement learning”. In: *Machine learning proceedings 1994*. Elsevier, pp. 157–163.

Lowe, Ryan et al. (2017). “Multi-agent actor-critic for mixed cooperative-competitive environments”. In: *Advances in neural information processing systems*, pp. 6379–6390.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, et al. (2013). “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602*.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, et al. (2015). “Human-level control through deep reinforcement learning”. In: *nature* 518.7540, pp. 529–533.

OpenAI (2018). *OpenAI Five*. `https://blog.openai.com/openai-five/`.

Otero, Daniel F and Akhavan-Tabatabaei, Raha (2015). “A stochastic dynamic pricing model for the multiclass problems in the airline industry”. In: *European Journal of Operational Research* 242.1, pp. 188–200.

Rana, Rupal and Oliveira, Fernando S (2015). “Dynamic pricing policies for interdependent perishable products or services using reinforcement learning”. In: *Expert systems with applications* 42.1, pp. 426–436.

Ross, Sheldon M (2014). *Introduction to stochastic dynamic programming*. Academic press.

Schulman, John et al. (2017). “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347*.

Shihab, Syed Arbab Mohd et al. (2019). “Autonomous Airline Revenue Management: A Deep Reinforcement Learning Approach to Seat Inventory Control and Overbooking”. In: *arXiv preprint arXiv:1902.06824*.

Shoham, Yoav, Powers, Rob, and Grenager, Trond (2003). “Multi-agent reinforcement learning: a critical survey”. In: *Web manuscript* 2.

Silver, David et al. (2017). “Mastering the game of go without human knowledge”. In: *nature* 550.7676, pp. 354–359.

Sims, Karl (1994). “Evolving virtual creatures”. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 15–22.

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

Tan, Ming (1993). “Multi-agent reinforcement learning: Independent vs. cooperative agents”. In: *Proceedings of the tenth international conference on machine learning*, pp. 330–337.

Vinyals, Oriol et al. (2019). “Alphastar: Mastering the real-time strategy game starcraft ii”. In: *DeepMind blog*, p. 2.

Wang, Rui et al. (2019). “Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions”. In: *arXiv preprint arXiv:1901.01753*.

Yaeger, Larry (1994). “Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or Poly World: Life in a new context”. In: *Santa Fe Institute*

Studies in the Sciences of Complexity-Proceedings. Vol. 17. Addison-Wesley Publishing Co, pp. 263–263.

Zhang, Dan and Cooper, William L (2009). "Pricing substitutable flights in airline revenue management". In: *European Journal of Operational Research* 197.3, pp. 848–861.

Zhang, Kaiqing, Yang, Zhuoran, and Başar, Tamer (2019). "Multi-agent reinforcement learning: A selective overview of theories and algorithms". In: *arXiv preprint arXiv:1911.10635*.

Appendix A Environment Details

A.1 Environment Parameters

The summarized version of all chosen simulation environment parameters can be seen below:

Airline Agents	1	Lower income no-show %	0.1
Customer Agents	[2,20]	Middle income no-show %	0.1
Timesteps	100	Upper income no-show %	0.2
Ticket Classes	3	No. 1st class tickets	2
Cancellation cost rate	0.2	No. 2nd class tickets	3
Bump cost rate	0.3	No. 3rd class tickets	5
Lower income %	0.3	1st class fixed cost	0.3
Middle income %	0.6	2nd class fixed cost	0.2
Upper income %	0.1	3rd class fixed cost	0.1
Lower income capital	(0.25,0.5]	1st class utility	1.0
Middle income capital	(0.5,0.75]	2nd class utility	0.75
Upper income capital	(0.75,1.0]	3rd class utility	0.5

A.2 Performance Comparison

In this portion of Appendix A, we describe, in detail, the inner workings of our performance comparisons found in Figure 6—namely the *ideal expected avg* line and the approximate dynamic programming approach.

A.2.1 Ideal Expected Average

To calculate the *ideal expected avg* dotted line from Figure 6a, we begin by calculating the expected amount of capital per customer. From the above table, let us define l_{pct}, m_{pct}, and u_{pct} as the lower, middle, and upper income % respectively. As we additionally uniformly sample the capital per customer class, let us also define $\hat{z}_l$, $\hat{z}_m$, and $\hat{z}_u$ to be the expected wealth of each customer class. Then the average expected capital per customer becomes

$$\hat{z} = l_{pct}\hat{z}_l + m_{pct}\hat{z}_m + u_{pct}\hat{z}_u.$$

With this knowledge, we can calculate the ideal amount of capital that the airline can obtain assuming it sells each ticket at price $\hat{z}$ through

$$\textit{ideal expected avg} = \begin{cases} \hat{z}(\text{No. Customers}) - (\text{Fixed investment cost}) & \text{if No. Customers} \leq S \\ \hat{z}S - (\text{Fixed investment cost}) & \text{else} \end{cases}$$

where S denotes the total ticket supply (No. 1st class tickets + No. 2nd class tickets + No. 3rd class tickets).

A.2.2 Approximate Dynamic Programming

Approximate Dynamic Programming (ADP) and RL are closely related in the sense that both attempt to optimize the Bellman equation. However, they differ in the sense that ADP requires full knowledge of the environment such as transition probabilities and reward functions beforehand whereas RL attempts to learn these things through sampling experiences from the environment. From this perspective, if all knowledge is known beforehand, it is much more efficient to use ADP than RL. However, the curse of dimensionality still applies—even if all knowledge is known beforehand, an exact solution quickly becomes infeasible as the state space grows and thus an approximate solution is used instead. In MARL, the state and action spaces quickly grow with the number of agents and thus we additionally include the ADP solution to compare how well it performs in comparison to our approach. Although an inherently uneven approach with one having access to complete environment information and the other not, we consider it a suitable benchmark due to the restriction of a large state space.

To significantly reduce the problem size and thus speed up computation time, we set the ADP algorithm to only optimize the airline's strategy. As a result, the ADP algorithm bases its optimization on the optimized MASAC customer agents. Taking inspiration from the stochastic pricing model of Otero and Akhavan-Tabatabaei (2015), our ADP solution has the task of solving the following objective function in each time period t:

$$\max_{\mathbf{a}_a} r_{t,a}(s_{t,a}, s_{t,c}, \mathbf{a}_a, \mathbf{a}_{t,c}) + \mathbb{E}_{\substack{s_{t+1,a}\sim\mathcal{I}(z_t)\\ s_{t+1,c}\sim\mathcal{I}(\mathbf{a}_a)\\ \mathbf{a}_{t+1,c}\sim\pi_{\theta_c}(s_{t+1,c})}} \Big[Q(s_{t+1,a}, s_{t+1,c}, \mathbf{a}_a, \mathbf{a}_{t+1,c})\Big], \tag{19}$$

where $Q(\cdot)$ is the multi-agent state-action value function from Subsection 3.4. For our MASAC algorithm, we compute the expectation by means of a Q-function approximator. However, since the ADP algorithm has access to all environment information, we calculate this expectation by means of a Monte-Carlo simulation that iterates through the simulation several times. In our implementation, we choose to iterate through the simulation 30 times. For a more concise view of this ADP optimization approach in action, we provide Algorithm 3.

Algorithm 3 Airline ADP Simulation

```
  Input: MASAC customer strategies $\pi_{i,c}$ for $i = 1, \ldots, C$ customers;
  1. Randomize environment settings using chance agent policy $\pi_{chance}$;
  2. Set time period $t \leftarrow 0$;
  3. Set agent $\leftarrow$ airline;
  4. Set $\mathrm{p}_{j,t} = \mathrm{b}_{j,t} \leftarrow 0$ for $j = 1, \ldots, J$ ticket classes;
  5. repeat
  6.     if agent = airline do
  7.         Get $\mathbf{a}_{t,a} = [p_{1,t}, \ldots, p_{j,t}, b_{1,t}, \ldots, b_{j,t}]^T$ from Equation 19;
  8.         Set environment prices/booking limits as $p_{j,t}$ and $b_{j,t}$;
  9.         Get new history $h_t \in \mathcal{H} \setminus \mathcal{Z}$ and information state $s_{t,c} = \mathcal{I}(\mathbf{a}_{t,a}) = \mathcal{I}(h_t)$;
 10.         Set agent $= \tau(h_t) \leftarrow$ customer;
 11.     end if
 12.     if agent = customer do
 13.         Get joint action $\mathbf{a}_{t,c} = [x_{1,t}, \ldots, x_{C,t}]^T$ from $\pi_{i,c}$ using information state $s_{t,c} \in \mathcal{S}$;
 14.         Process customer $i$'s action in environment;
 15.         Get new history $z_t \in \mathcal{Z}$ and information state $s_{t,a} = \mathcal{I}(z_t)$;
 16.         Get airline and customer rewards $[r_{t,a}(z_t), r_{1,t,c}(z_t), \ldots, r_{C,t,c}(z_t)]^T$;
 17.         Set agent $= \tau(z_t) \leftarrow$ airline;
 18.         Set $t \leftarrow t + 1$;
 19.     end if
 20. until $t = T$
```

Appendix B Optimization Implementation Details

B.1 Neural Net Architecture

For both the airline and customers agents, the policy π and Q-function ϕ neural networks follow similar constructions except for the final output layers. For both π and ϕ, we have the input observations pass through two fully connected dense layers. Between each layer, the respective outputs first undergo a ReLU activation and then are subsequently normalized. Once here, the outputs go through a final dense layer with π_a adding a sigmoidal activation to the 6 airline actions and π_c adding a softmax activation to the 4 customer actions. Each ϕ outputs a single value with no activation. For all layers, we employ both a L1 and L2 weight regularization with the regularizing coefficient set to be 0.01. We additionally set the size of each of the dense layers to have 128 nodes (excluding the output dense layers).

B.2 Hyperparameters

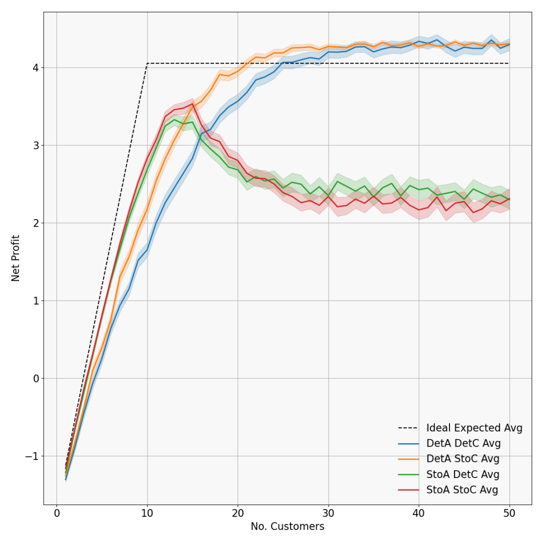
Table 1: Optimization Hyperparameters

Parameter	Value
Random seed	0
Optimizer	Adam (Kingma and Ba, 2014)
Total environment steps	1.4 million
Learning rate	0.001
Discount rate (γ)	0.99
Replay buffer size	20000
Polyak coefficient (ρ)	0.005
Minibatch size	1028
No. update steps	20
Steps per update	4000
Target entropy (airline)	$[-1, -1, -1, -1, -1, -1]^T$
Target entropy (customer)	0.98 * (-log(1/4)) (Christodoulou, 2019)
Initial α	0.2

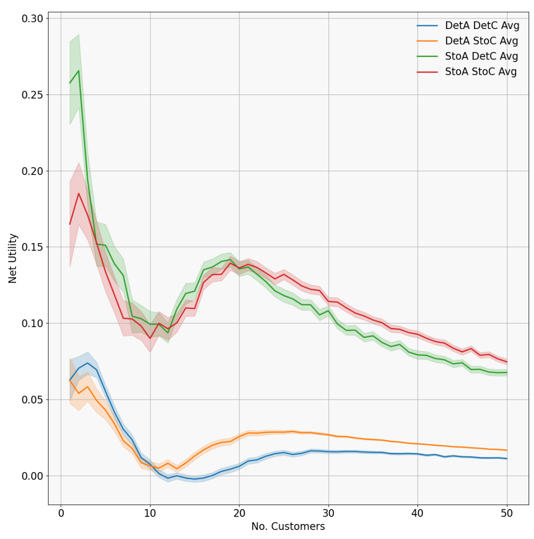
Appendix C Extended Results

In our research, we trained our airline and customer agents on simulations with a randomly initialized number of customers between 2 and 20. Thus, all subsequent results shown in Section 6 were limited to a maximum of 20 customers. In this section, we extend those results by having the airline and customer agents interact in simulations where there are more than 20 customers—which the agents have not trained in. Specifically, we provide results for 1 to 50 customers.

From the Figures shown below, we see a more complete picture of how the airline and customer agents perform in varying situations. For example, in Figure 14, we see that for > 20 customers, the deterministically sampling airline outperforms the *ideal expected avg* benchmark but eventually converges in its performance as the number of customers grow. Naturally, this convergence just above the *ideal expected avg* benchmark is a result of the profit gained from risky overbooking: as the number of customers increase, the demand concentration strategy becomes more likely to be successful. Figure 15b adds onto this as we see that the number of over-bookings converges to 1 as the number of customers increase. This convergence can similarly be seen for the customer net utility. However, there remains a slight downward trend in Figure 14b as the increasing number of customers naturally means more customers not having a ticket (and thus a net utility of 0) which pushes the average net utility downwards.

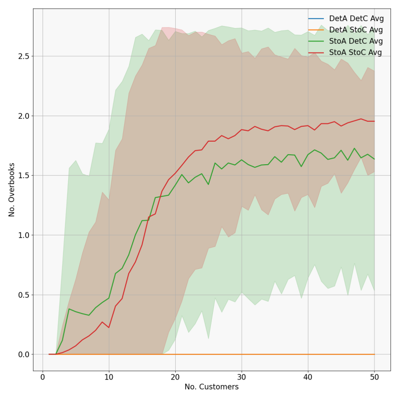


(a) Airline Avg Net Profit

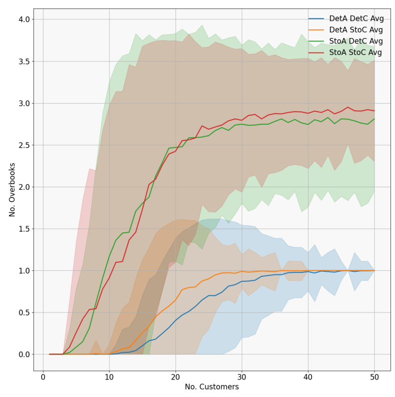


(b) Customer Avg Net Utility

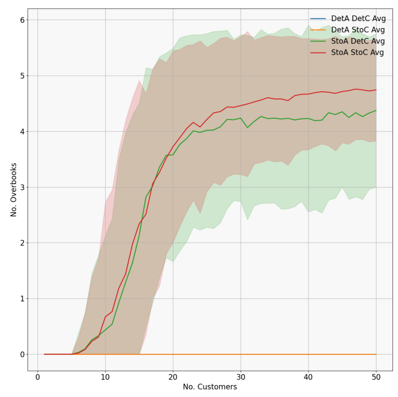
Figure 14: Net Profit/Utility
This figure shows the average net profit and utility (and their 95% confidence intervals) obtained by the airline (A) and customer (C) agents respectively for 1 to 50 customers. We additionally include the results for different combinations of the action sampling schemes (deterministically (Det) and stochastically (Sto)). Thus for 14a, the initial investment cost of -1.7 is automatically included. The averages and confidence intervals are obtained through 300 simulation runs per number of customers. For the airline net profit, we additionally include an *ideal expected avg* line that plots the expected net profit in the idealized situation where the airline can sell every ticket at a price equal to the expected amount of capital per customer. This line does not consider the potential profit gain from overbooking tickets.



(a) 1st Class



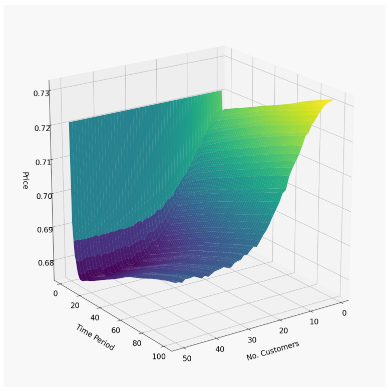
(b) 2nd Class



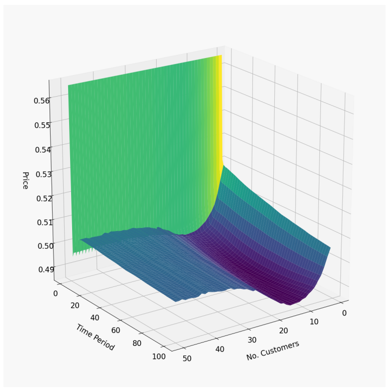
(c) 3rd Class

Figure 15: Average No. Over-bookings per Ticket Class
This figure consists of plots of the average number of tickets overbooked per ticket class for each MASAC sampling scheme. DetA and StoC stand for when the airline samples actions deterministically and customers sample actions stochastically respectively. Besides showing the average, the plots additionally show the standard deviation of the number of bookings incorporating approximately 96% of the observations. The averages and standard deviations are calculated based on 300 simulation runs per customer size.

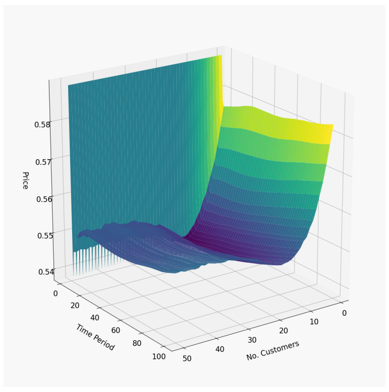
If we additionally consider the strategies that the airline takes in Figures 16 and 17, we also see a more well-rounded picture where the previously seen U-shaped strategies do not actually possess a U-shape. This only provides further evidence to the observation that the airline's policy is capable of extending past what it has trained on. If the strategies in Figures 7 and 8 were actually U-shaped past 20 customers, the airline performance would naturally suffer when faced with > 20 customers.



(a) 1st Class

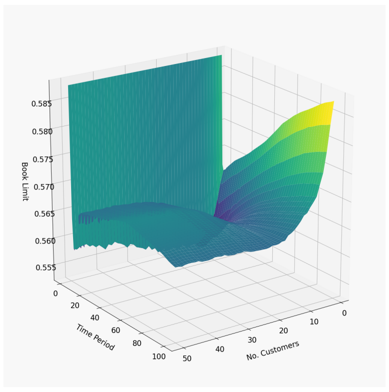


(b) 2nd Class

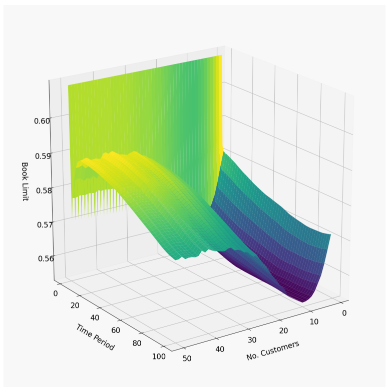


(c) 3rd Class

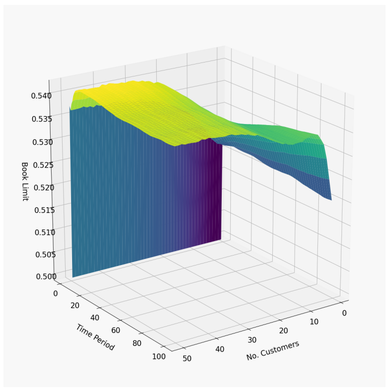
Figure 16: Average Price per Ticket Class
This figure consists of surface plots of the average price for each ticket class over time and number of customers. The averages are calculated based on 300 simulation runs per customer size where the airline and customer agents' actions were deterministically sampled.



(a) 1st Class



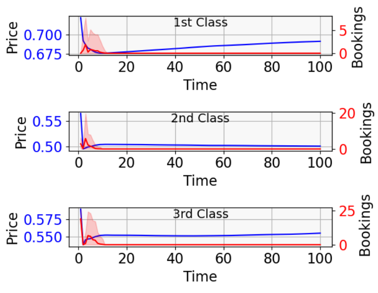
(b) 2nd Class



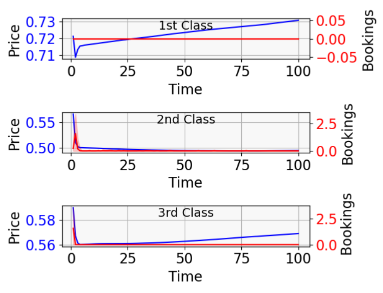
(c) 3rd Class

Figure 17: Average Booking Limit per Ticket Class
This figure consists of surface plots of the average booking limit for each ticket class over time and number of customers. The averages are calculated based on 300 simulation runs per customer size where the airline and customer agents' actions were deterministically sampled.

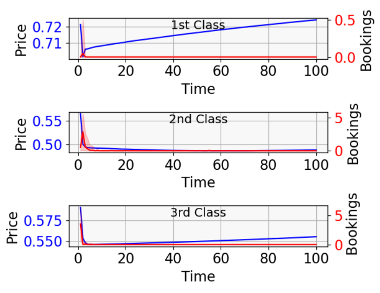
Lastly, Figures 18 and 19 have been extended to show the average booking distributions for a varying number of customer sizes. These results only serve to enhance the comments made in Section 6. We additionally include Figures that show the average number of cancellations for each customer class and a varying number of customer sizes.



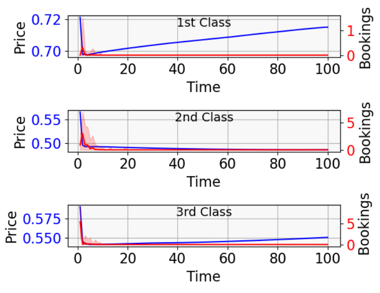
(a) 1 Customer



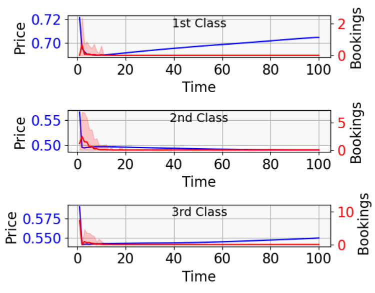
(b) 5 Customers



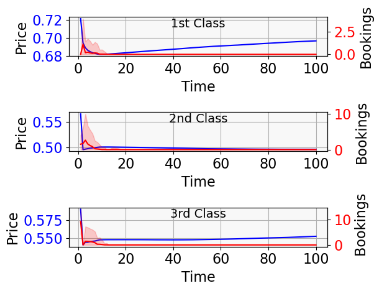
(c) 10 Customers



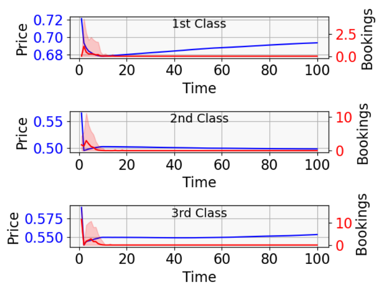
(d) 15 Customers



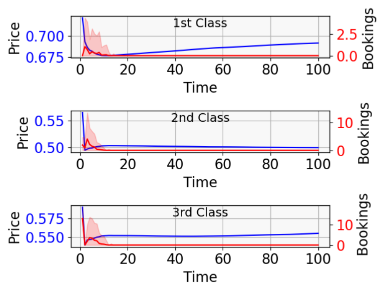
(e) 20 Customers



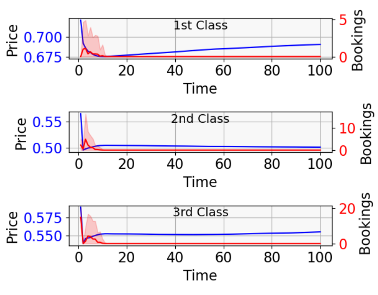
(f) 25 Customers



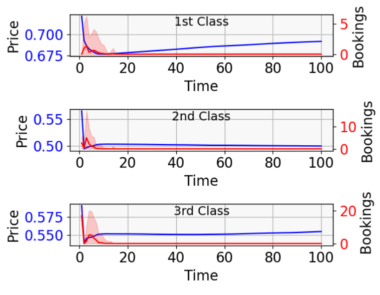
(g) 30 Customers



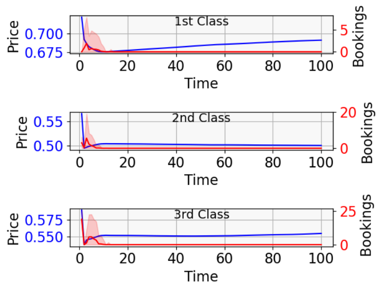
(h) 35 Customers



(i) 40 Customers

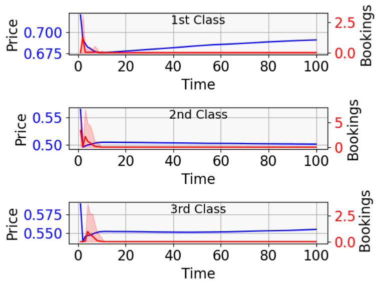


(j) 45 Customers

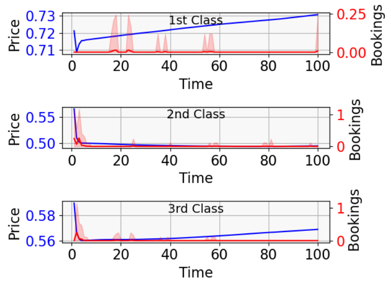


(k) 50 Customers

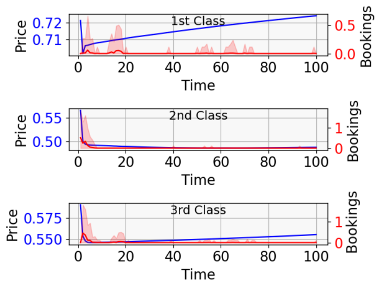
Figure 18: Average No. Bookings for Middle Class Customer
This figure consists of plots of the average price compared to the average number of bookings for customers who are initialized to be middle class. Besides showing the average, the plots additionally show the standard deviation of the number of bookings incorporating approximately 96% of the observations.



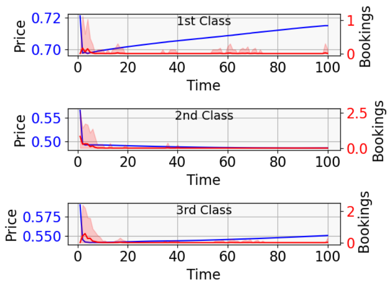
(a) 1 Customer



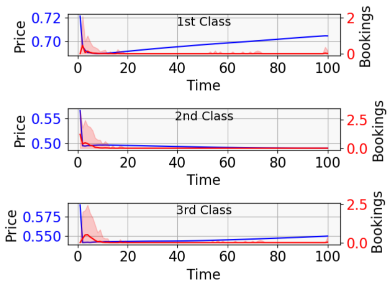
(b) 5 Customers



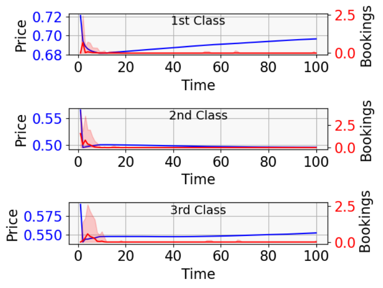
(c) 10 Customers



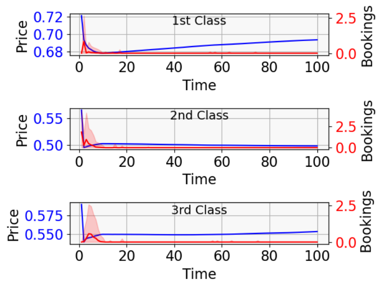
(d) 15 Customers



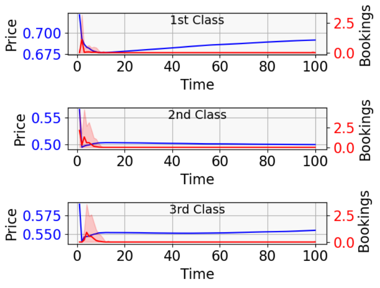
(e) 20 Customers



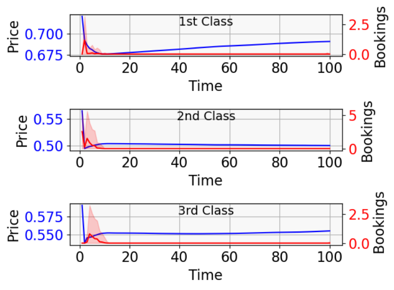
(f) 25 Customers



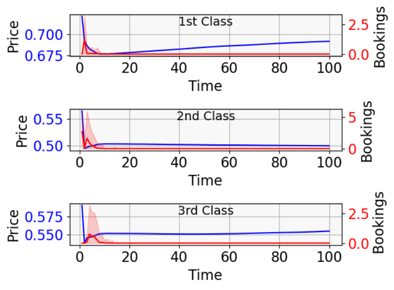
(g) 30 Customers



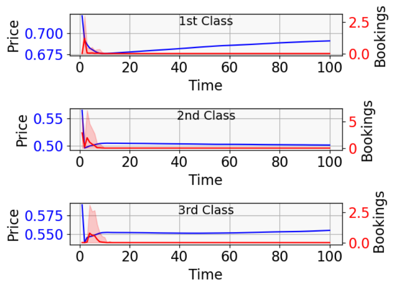
(h) 35 Customers



(i) 40 Customers

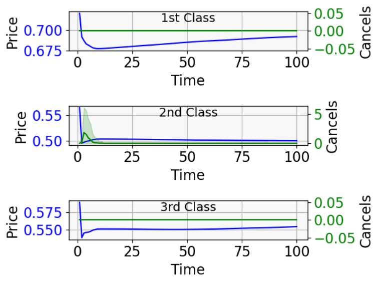


(j) 45 Customers

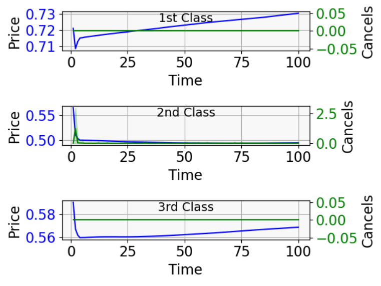


(k) 50 Customers

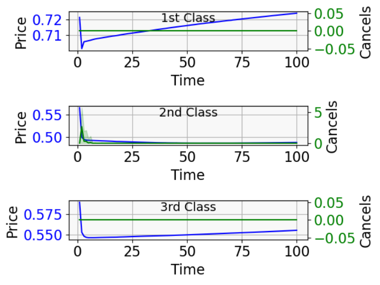
Figure 19: Average No. Bookings for Upper Class Customer
This figure consists of plots of the average price compared to the average number of bookings for customers who are initialized to be upper class. Besides showing the average, the plots additionally show the standard deviation of the number of bookings incorporating approximately 96% of the observations.



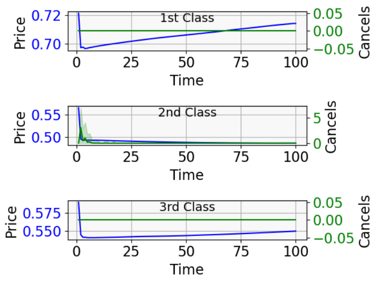
(a) 1 Customer



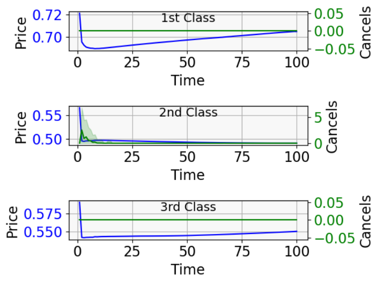
(b) 5 Customers



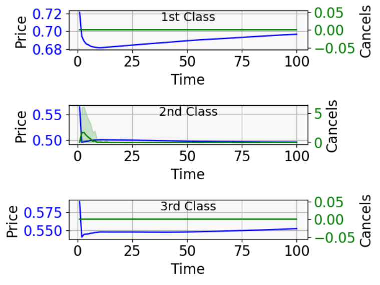
(c) 10 Customers



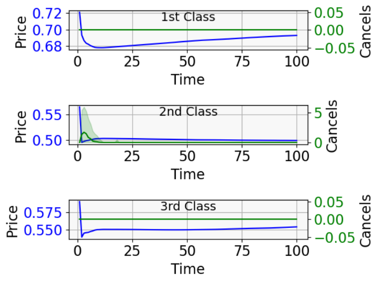
(d) 15 Customers



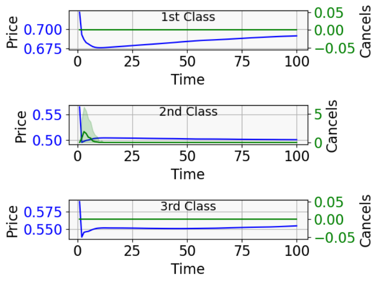
(e) 20 Customers



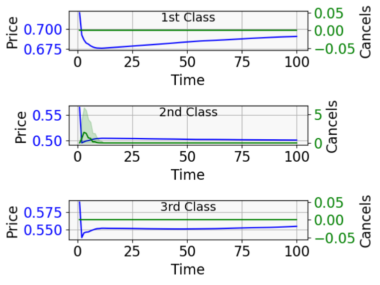
(f) 25 Customers



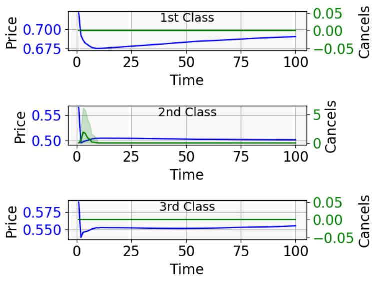
(g) 30 Customers



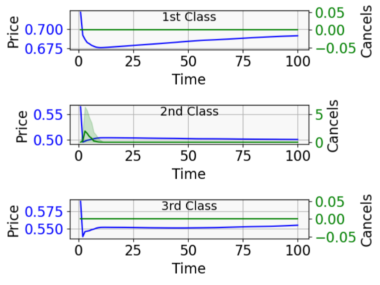
(h) 35 Customers



(i) 40 Customers

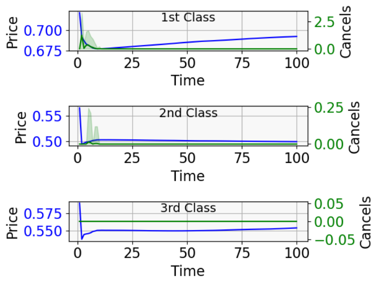


(j) 45 Customers

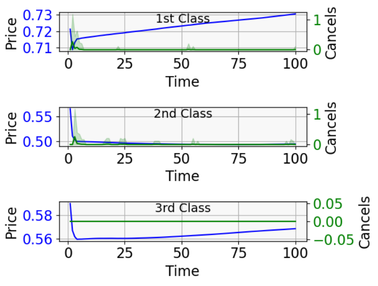


(k) 50 Customers

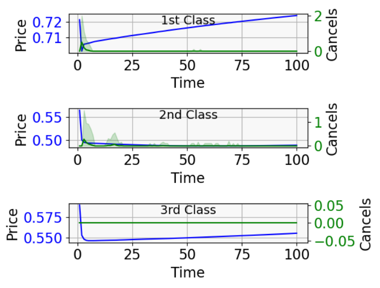
Figure 20: Average No. Cancellations for Middle Class Customer
This figure consists of plots of the average price compared to the average number of ticket cancellations for customers who are initialized to be middle class. Besides showing the average, the plots additionally show the standard deviation of the number of cancellations incorporating approximately 96% of the observations.



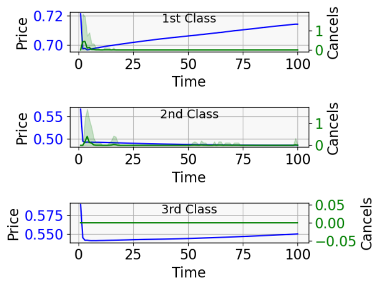
(a) 1 Customer



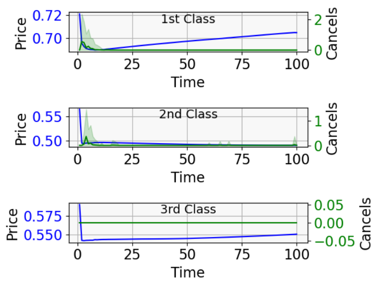
(b) 5 Customers



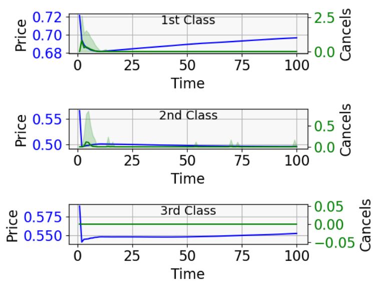
(c) 10 Customers



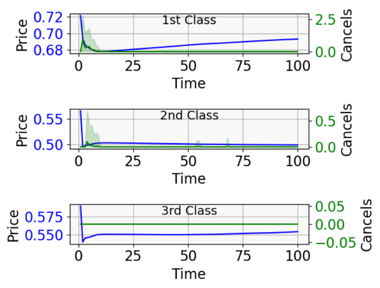
(d) 15 Customers



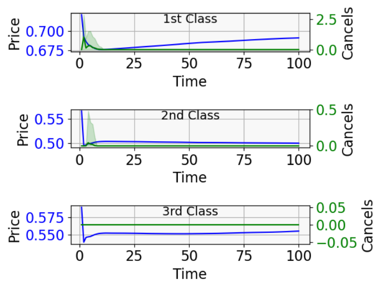
(e) 20 Customers



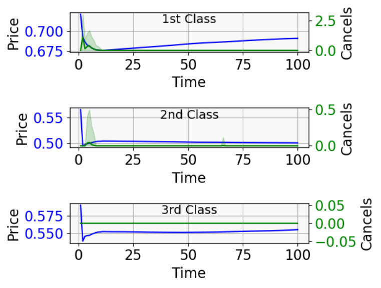
(f) 25 Customers



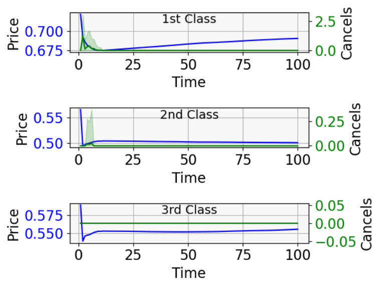
(g) 30 Customers



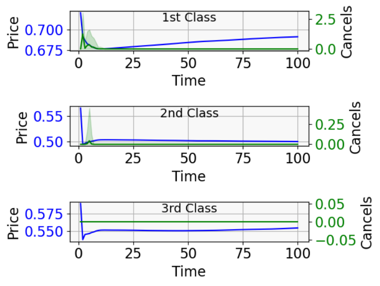
(h) 35 Customers



(i) 40 Customers



(j) 45 Customers



(k) 50 Customers

Figure 21: Average No. Cancellations for Upper Class Customer
This figure consists of plots of the average price compared to the average number of ticket cancellations for customers who are initialized to be upper class. Besides showing the average, the plots additionally show the standard deviation of the number of cancellations incorporating approximately 96% of the observations.