# Multi-Fare Airline Ticket Dynamic Pricing Using Model-Free Reinforcement Learning

Group 117 - Final Project Status Update
Ross B. Alexander, Jonathan S. Ling

## Background

Dynamic pricing in the airline industry demonstrates some of the most effective pricing schemes in business to maximize revenue based on customers' willingness to pay for particular goods at particular times. We propose to develop a dynamic pricing reinforcement learning algorithm to maximize revenue for a single flight with multiple customer segments. We suggest reinforcement learning as it is a model-free paradigm and thus less sensitive to unusual demand patterns, and because it is a relatively new approach to dynamic pricing for airlines.

This is a decision-making problem under uncertainty that has two distinct perspectives - airline and customer. The airline must determine what prices to set without knowing the underlying stochastic demand functions of customers. In seeking to maximize revenue, the airline must balance the need to sell all their seats before the departure date with waiting until the last minute to sell their most expensive seats. The customer, in seeking to obtain a single ticket at the lowest cost, has the opportunity to learn from the advertised prices over time in order to estimate the best time to purchase a ticket, but must also purchase a ticket before all tickets are gone.

## Problem Formulation

We formulate a Markov decision process (MDP) with the following 'sub-MDPs' for each fare class $f \in \mathcal{F} = \{1,2,3\}$ as follows:

- States $(v,u,t) \in \mathcal{S}_f$ are composed of the total number of tickets vacant (available) $v \in [0,V]$, the number of tickets utilized (sold) in the $f$ fare class $u \in [0,U_f]$, and the current time step $t \in [0,T]$
- Actions $a \in \mathcal{A}_f$ are ticket price assignments for the corresponding fare class, where each fare class price list $\mathcal{A}_f$ contains 30 prices in increments of \$10 with the upper and lower price bounds set by the fare class
- Reward $r \in \mathcal{R}$ is the revenue, which is the sum over all fare classes of ticket price times quantity sold at each time step

We then solve an overarching MDP that summarizes all fare classes together. It differs from the fare class-specific MDPs in the following ways:

- States $(v,t) \in \mathcal{S}$ have no $u$ variable
- Actions $(a_1, a_2, ..., a_{|\mathcal{F}|}) \in \mathcal{A}$ are now vectors of size $|\mathcal{F}|$ to denote the prices set by each fare class

Note that all seats are equal (e.g., all economy), and thus available inventory is from a common pool. However, we can distinguish between fare classes and thus charge different prices by the customer details provided (e.g., one-way/return, length of trip, etc.).

Also, in place of a dataset, we formulate a generative model $(s', r) \sim G_f(s,a)$, to simulate the demand and purchase of tickets in each fare class $f$.

For this, we define the following quantities:

- $n$ is the number of new customers at the current time step
- $C_f$ is the (initially empty) set of all customers in fare class $f$ seeking tickets at the current time step, i.e., new customers from past time steps that did not purchase, plus all new customers from the current time step. (We initialize this in Algorithm 2 as a global variable)
- $w_i$ is customer $i$'s 50% willingness-to-pay (WTP) threshold
- $k_i$ is customer $i$'s WTP flexibility
- $\phi_i$ is customer $i$'s likelihood of buying at price $a$
- $b_i \in \{0,1\}$ indicates whether customer $i$ in fare class $f$ bought the ticket (1) or not (0)

We use Algorithm 1 as our generative model. At each time step, we select some number of new customers from a Poisson distribution (with a linearly decreasing mean), randomly generate their WTP characteristics, and from this, determine how many tickets get sold.

**Algorithm 1** GENERATIVEMODEL$_f(s(v,u,t),a)$

```
1:  $u' \leftarrow 0$
2:  $n \sim \text{Poisson}(\alpha_f \cdot (T-t) + \beta_f)$ for some constants $\alpha_f, \beta_f$
3:  for $i$ in 1 to $n$ do
4:      $w_i \sim \mathcal{N}(\mu_{wf}, \sigma_{wf})$
5:      $k_i \sim \mathcal{N}(\mu_{kf}, \sigma_{kf})$
6:      $C_f \leftarrow C_f \cup \{i\}$
7:  end for
8:  for $i \in C_f$ do
9:      $\phi_i = 1 - \text{CDF}(\text{Logistic}(a \mid w_i, k_i))$
10:     $b_i \sim \text{Bernoulli}(1, \phi_i)$
11:     if $b_i = 1$ then
12:         $u' \leftarrow u' + 1$
13:         $C_f \leftarrow C_f \setminus \{i\}$
14:     end if
15: end for
16: $r \leftarrow a \cdot u'$
17: return $((v-u', u', t+1), r)$
```

We have chosen distributions over each of the random variables in a way that represents the problem dynamics. In some cases, such as the customer's WTP threshold and WTP flexibility, alternative distributions can also be justified. The customer's WTP is modeled as a complementary cumulative distribution function for the logistic distribution. The following figures depict the customer pricing dynamics and customer arrival dynamics.

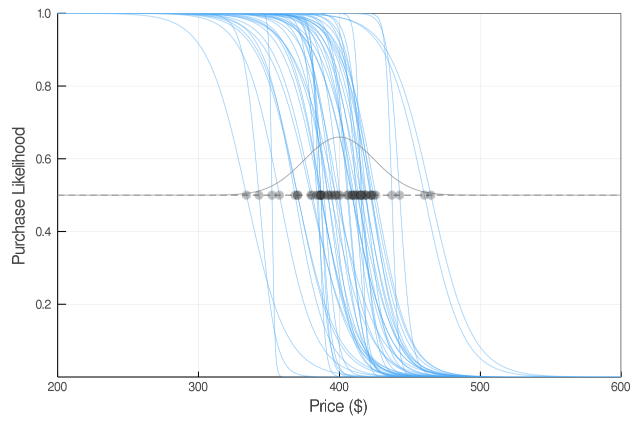

**Figure 1** 50 samples of customer purchasing schedules

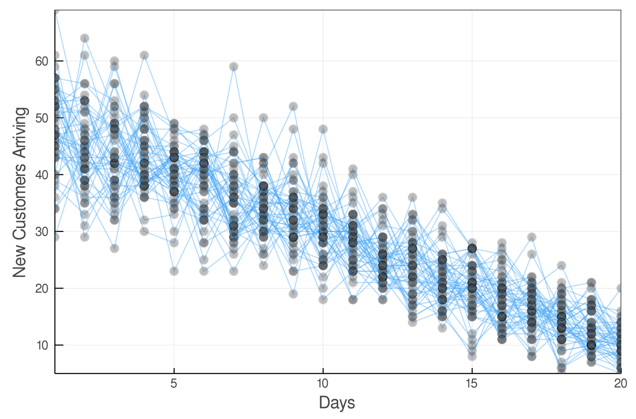

**Figure 2** 50 samples of the arrival of customers over a 20-day horizon

We solve the overarching MDP using the Sarsa method, as shown in Algorithm 2. At each time step, we call the generative model for each fare class and add their contributions to get the next state $(v', t')$ and current reward $r$, then generate the next action $a'$ and use the Sarsa update rule.

In line 10, we check to see if we have oversold our stock of tickets. If so, we correct it in line 11 by simulating selling the remaining tickets to customers on a first-come, first-served basis. E.g., if 15 customers wanted to buy from a remaining pool of 5 tickets, we would randomly pick 5 of the customers, independent of fare class, to obtain those tickets.

**Algorithm 2** SOLVEMDP

```
1: $C_f \leftarrow \emptyset, \forall f \in \mathcal{F}$
2: $(v, t) \leftarrow (V, T)$
3: $r \leftarrow 0$
4: Initialize $Q(s, a) = 0, \forall s \in S, a \in A$
5: Select $a$ using $\epsilon$-greedy with Gaussian randomness
6: repeat
7:     for $f \in \mathcal{F}$ do
8:         $(v', u'_f, t'_f), r_f \leftarrow \text{GENERATIVEMODEL}_f((v, 0, t), a)$
9:     end for
10:    if $\sum_f u'_f > v$ then
11:        Randomly choose values for each $u'_f$ so that $\sum_f u'_f = v$
12:    end if
13:    $(v', t') \leftarrow (v - \sum_f u'_f, t + 1)$
14:    $r \leftarrow \sum_f a_f \cdot u'_f$
15:    Select $a'$ using $\epsilon$-greedy with Gaussian randomness
16:    $Q(s, a) \leftarrow Q(s, a) + \eta(r_t + \gamma Q((v', t'), a') - Q(s, a))$
17:    $s \leftarrow (v', t'), a \leftarrow a'$
18: until $t = T$ or $v' = 0$
```

Finally, we can extract the optimal policy, $\pi^*$ using Algorithm 3.

**Algorithm 3** GETPOLICY($Q$)

```
1: for $s \in S$ do
2:     $\pi^*(s) \leftarrow \arg\max_a Q(s, a)$
3: end for
4: return $\pi^*$
```

## Timeline to Completion

First, we will continue to develop the generative model and then solve a single fare MDP. Following this, we will move to the multi-fare MDP formulation. We are using Sarsa, and will later Sarsa($\lambda$) to approximate the value function, though local or global approximation methods may be exploited later to improve generalization of the agent. We are on track to complete everything by December 6th.