Assignment 1

Haohui Zhang(2722930),Yongqing Liang(2721589)

November 14, 2021

1 Introduction

In this assignment, two models are developed, the first model is used to find the optimal policy and the total expected revenue, the second model is based on the first model with the addition of the restriction that the price cannot be lowered. Finally, we compare the total expected revenue of the two models using the corresponding policies and draw conclusions.

2 Question a

2.1 Scope limitation

- t is the timestamp of the current state. $t \in \{1, \ldots, T+1\}$
- c is the number of seats left in the current state. $c \in \{0, \ldots, C\}$

2.2 Some prerequisites are specified

- At most there is one person buying one seat per timestamp
- $\Sigma_i \lambda_t(i) \leq 1$ for all t
- Probability of demand depands on class i and time t: $\lambda_t(i)$

2.3 Some boundary conditions

- when c=0, then the ticket price is 0, $p_t(c \mid c, a) = 1$ means $p_t(0 \mid 0, 0) = 1$
- T+1 is departure moment so that:

$$V_{T+1}(c) = 0 \quad \forall c \in C \tag{1}$$

2.4 Basic ideas:

1. Ticket prices could be 500,300,200. Each Action is denoted by a
2. After that, every customer has a willingness-to-pay list $f \in \{500, 300, 200\}$. Depending on s, each price in the price list has a different probability. So we can conclude that:

$$p_t(c-1 \mid c, a) = \sum_{i=1}^{a} \lambda_t(i) \tag{2}$$

$$p_t(c \mid c, a) = 1 - \sum_{i=1}^{a} \lambda_t(i) \tag{3}$$

3. Next, we calculate the revenue of each pathway and select the Action with the largest revenue. The state transfer equation is shown below. After calculation, the total expected revenue is 30457.82.

$$V_t(c) = \max_{a=1,\ldots,n} \left\{ \sum \lambda_t(i) \left(f_a + V_{t+1}(c-1) \right) + \left(1 - \sum_{i=1}^{a} \lambda_t(i) \right) V_{t+1}(c) \right\} \tag{4}$$

4. When calculate the max revenue given timestamp t and capacity c, we get a index a, this is the optimal policy in the current state.

3 Question b

A plot of the optimal policy is 1:(Yellow is priced at 500, green at 300 and blue at 200)

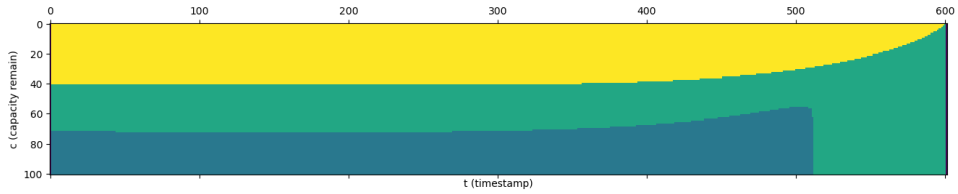


Figure 1: Optimal Policy

4 Question c

In this question, we need to simulate the demand and find the result of how many tickets are sold, the remaining capacity and the price at each moment. We determine demand distribution according the λ which is the probability of request depends on class and time. The equation as follow: $\lambda_t(i) = \mu_i e^{v_i t}$. The total time is 600, and at most 1 request per period, so we need to simulate 600 demands. For each time, we will get 3 probabilities correspond to each bid separately 500, 300, 200 by the customer. Because the $\sum_i \lambda_t(i) \leq 1$, the customer still have the probability to request a 0 demand. So based on the probability of each time, we sample the demand randomly. Now we get the 600 demands, we need to simulate the actions according the optimal policy we got in question a. The pesudo-code of algorithm 1 is clarified in appendix. The simulation result is in Figure 2. For this realization, the total reward is 33000, the remaining capacity is 0, the last ticket are sold at time 594.

	0	1	2	3	4	5	6	7	...	586	587	588	589	590	591	592	593
time	1	2	3	4	5	6	7	8	...	587	588	589	590	591	592	593	594
states	100	100	100	100	100	100	100	100	...	6	6	5	4	3	2	1	1
actions	200	200	200	200	200	200	200	200	...	500	500	500	500	500	500	500	500
demand	0	0	0	0	0	0	0	0	...	300	500	500	500	500	500	300	500
rewards	0	0	0	0	0	0	0	0	...	0	500	500	500	500	500	0	500

[5 rows x 594 columns]

total reward: 33000

Figure 2: Simulations

5 Question d

In this question, we need to make the price cannot go down. Based on the theory of the optimal policy generation we discussed on question a, there is only two situations:

- If the deal doesn't go through on the time t, which suppose price[x][t] in the optimal policy, the next the action made by airline company should be request the price on [x][t+1].
- If the deal successfully goes through on the time t, which suppose price[x][t] in the optimal policy, the next the action made by airline company should be request the price on [x-1][t+1].

In order to fulfill this, we need to assure the optimal policy matrix:

- The value of policy on the right is not less than the value on its left, which is price[x][t+1] $\geq$ price[x][t].
- The value of policy is not less than the value below its left, which is price[x-1][t+1] $\geq$ price[x][t].

Based on the above two criteria and the equation 4, we add state component to remember last price, compare it with the present state price and modify our expected value algorithm. The algorithm is presented in 2 and the plot of the new policy is shown in 4 in the appendix. According the new policy matrix, we recalculate the matrix using the equation 4 to verify the accuracy of our new expected value function. Now the final expected revenue decreased from 30457.8213 to 29573.1396. This is because the expected revenue obtained from question a is the best expected revenue in this situation, in order to pursue the price-not-go-down principle, the optimal policy is modified and the expected revenue will definitely decreased. We used the new policy matrix to simulate the same demand with question c, the remaining capacity is 3 and the total revenue becomes 32900. The total revenue is not significantly decreased is that although the remaining capacity increased from 0 to 3, due to the changed policy, a few tickets are sold at higher prices after t=480(in this simulation). The price, states and actions on each moment is showed in Figure 5b.

6 Conclusion

We successfully implement two models, separately the original optimal policy and the total expected revenue model and the model which is based on the first model with the addition of the restriction that the price cannot be lowered. The total expected revenues of the two models are respectfully 30457.8213 and 29573.1396. Besides, we simulate two models with the same demand. We construct the demand matrix based on the probability. The comparison result is shown in appendix. We can clearly observe the difference of action between Figure 5a and 5b, in Figure 5b the price does not go down and because of that the success rate of transaction decrease. The total revenue of two simulations are 33000 and 32900.

A Appendix

Algorithm 1 Simulate

Require: List $Policy$ & List $Demand$
$time = 0, capacity = 100$
Initialize List $Result$
for $state = 99, 98 \ldots, 0$ **do**
  for $index, p$ in enumerate $Policy$ **do**
    if $index = time$ **then**
      if $time = 600$ or $capacity = 0$ **then**
        Break
      else if $Demand[time] \geq p$ **then**
        $Result \leftarrow p$
        $capacity = capacity - 1$
        $time = time + 1$
        Break
      else
        $Result \leftarrow 0$
        $time = time + 1$
      end if
    end if
  end for
end for

Algorithm 2 Price-never-go-down Algorithm

Require: f & μ & v
Initialize List $Value$ & List $Policy$
for $state = 1, 2, \ldots, 100$ **do**
  for $time = 599, 598, \ldots, 0$ **do**
    Calculate the Probability λ of each f at time t
    $Value \leftarrow \max(f * \lambda)$
    $Policy \leftarrow f$
    if $t < 599$ **then**
      if $Policy >$ last state $Policy$ **then**
        for All previous state $time = 599, \ldots, t$ **do**
          $Policy \leftarrow Policy[x - 1][t]$
          Recalculate the $Value$
        end for
      end if
    end if
  end for
end for

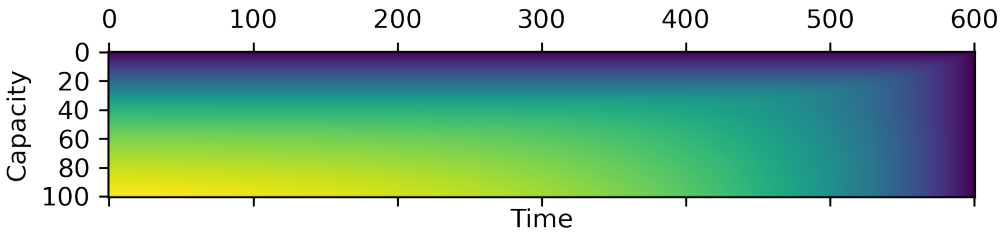


Figure 3: Plot of the new expected revenue

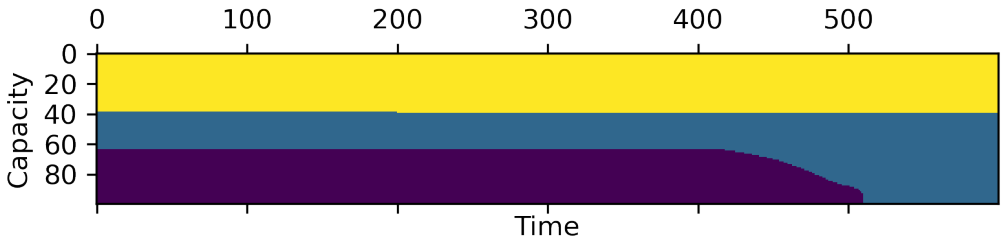
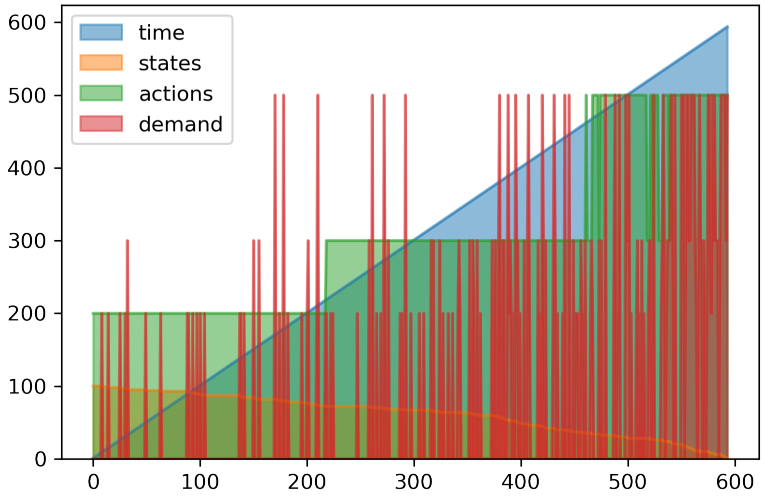
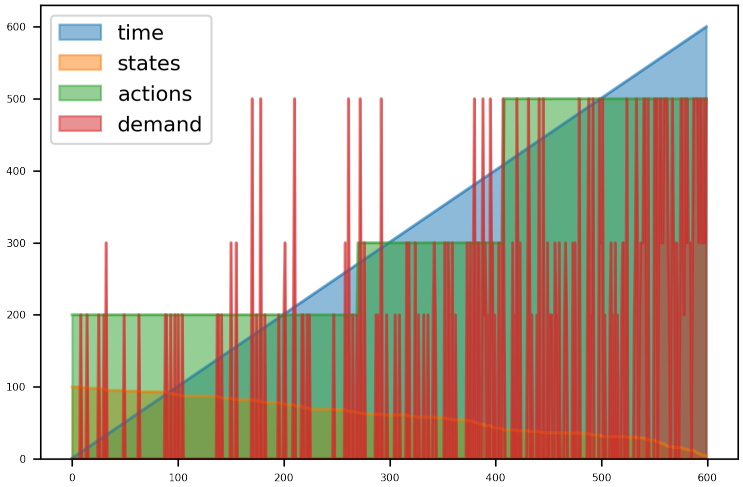


Figure 4: Plot of the new policy



(a) Simulations plot in question c



(b) Simulations plot with the same demand in question c

Figure 5: Comparison Plot of Question c&d