Optimization Project II:
Maximizing Revenue in the Skies: A Dynamic Approach to Overbooking

Austin Yeh, Felipe Zapater, Sonali Hornick, Sneha Sastry Rayadurgam

Introduction

Objective

We work for an airline company, leading its revenue management. The objective of this project is to develop an optimal overbooking and ticket pricing policy for a specific flight using the concepts of dynamic programming. Our end goal is to maximize the company's expected discounted profit over one year period by balancing ticket revenue with the overbooking costs, while adhering to constraints such as seat capacity and customer goodwill.

Airline Pricing and Overbooking

The concept of revenue management was first introduced by the airline industry. Apart from the competitors and the market conditions, a lot of other factors play an important role in pricing the airline tickets. There's some optimization in revenue by introducing tiers in the flight seats - creating product or service distinction - making it attractive for the less price sensitive customers to opt for higher priced seats for improved comfort. Another common practice in the revenue management vertical for this industry is overbooking. Determining the effectiveness of this policy requires careful analysis of revenue and cost trade offs. This project is an attempt to explore just that - should the airlines overbook tickets, and if so, by how many seats, in order to achieve highest possible profits. Using the concepts of dynamic programming, we would be optimizing the daily ticket pricing for coach and first class seats over a 365 day period. The project will evaluate various policies and assumptions by simulating passenger demand, show-up probabilities and the associated costs.

Key elements explored include

a. **Dynamic Ticket Pricing:** Adjusting coach and first-class ticket prices daily to balance demand fluctuations and maximize revenue.
b. **Overbooking Policy Optimization:** Exploring various overbooking scenarios to determine the number of coach seats that can be oversold without excessive penalties.
c. **Customer Behavior Modeling:** Incorporating probabilities of passenger attendance and purchasing decisions based on price sensitivity.
d. **Seasonality Effects:** Accounting for increased demand closer to departure dates.

Simplifications made

a. We have assumed that there are only two classes or types of seats available on the flight
b. We have assumed that there are only two possible prices that we could pick from for each of the tiers - in reality, there might be a bigger set of prices available to choose from
c. We have also assumed that only one ticket is sold per class on any given day. However, in reality, multiple tickets can be sold, and the probability of selling a certain number of seats would follow a discrete distribution that is not necessarily binomial.
d. We have not taken into account the effect of competitor pricing at any point in our solution. In reality, these factors also play a key role and change the probabilities associated with the sale of the tickets

Dynamic programming helps us break down the complex problem of revenue management and profit optimization into smaller problems and reusing these values, whenever needed. By solving this problem iteratively backward, the project identifies optimal strategies for pricing and overbooking under different scenarios and the improvement in profits by adopting these strategies.

Problem Set-up

Fundamentals of Dynamic Programming

The following are the key steps in dynamic programming:

1. Backward Calculation (Value Function Iteration): The value function for the final period is calculated based on the given state and decision variables, as there are no future periods to consider. This step uses the principle of optimality, which ensures that the solution for each stage depends only on the current state and decision
2. Recursive Computation (Bellman Equation): Using the calculated value function of the last period, earlier periods are solved recursively. The Bellman equation defines this process mathematically:

$$V_t(x_t) = \max_{u_t} \left[F(x_t, u_t) + \beta V_{t+1}(x_{t+1}) \right]$$

Here, *F*(*xt*,*ut*) represents the immediate reward or payoff from state *xt* and decision *ut*, while *Vt*+1(*xt*+1) accounts for future rewards

3. Forward Computation (Policy Derivation): After solving for all value functions backward in time, the optimal decisions (policies) for each state and time are derived by moving forward through the periods. This involves selecting actions at each stage that maximize the overall value function

Setting up the basic dynamic problem

For any given dynamic programming problem, there are six main components to establish

<u>State Variables</u>

These are the variables which would describe the state of the system at any given time. For our problem, the following are the state variables.

$$S=(t,c,f)$$

- $t \in \{0,1,\ldots,365\}$: Days remaining until departure
- $c \in \{0,1,\ldots,C_{max}\}$: Coach tickets sold ($C_{max}$=100+overbooking limit)
- $f \in \{0,1,\ldots,20\}$: First-class tickets sold

<u>Decision Variables</u>

These define the possible decisions we could take based on our state variables. In our case, the decision variable would be the price of the first class and the coach ticket for each of the day:

- *pc*: Price for coach tickets ($300 or $350)
- *pf*: Price for first-class tickets ($425 or $500)

<u>Dynamics</u>

The probability of selling seats in coach and first class is give as follows:

Coach Sale Probability

$$\text{prob}_c = \begin{cases} 0.65 + 0.03 \times 1_{\{f \geq 20\}} & \text{if} p_c = 300 \text{and} c < C_{\max} \\ 0.30 + 0.03 \times 1_{\{f \geq 20\}} & \text{if} p_c = 350 \text{and} c < C_{\max} \\ 0 & \text{otherwise} \end{cases}$$

The probability of purchasing a seat in coach increases by 3% when all the seats in the first class are sold

First-class Sale Probability

$$\text{prob}_f = \begin{cases} 0.08 & \text{if} p_f = 425 \text{and} f < 20 \\ 0.04 & \text{if} p_f = 500 \text{and} f < 20 \\ 0 & \text{otherwise} \end{cases}$$

We will then use these probabilities to write the dynamics for the problem.

- If a coach ticket is sold (sc =1), $c'=c+1$(up to Cmax)
- If a first-class ticket is sold (sf =1), $f'=f+1$(up to 20)
- Otherwise, $c'=c$, $f'=f$

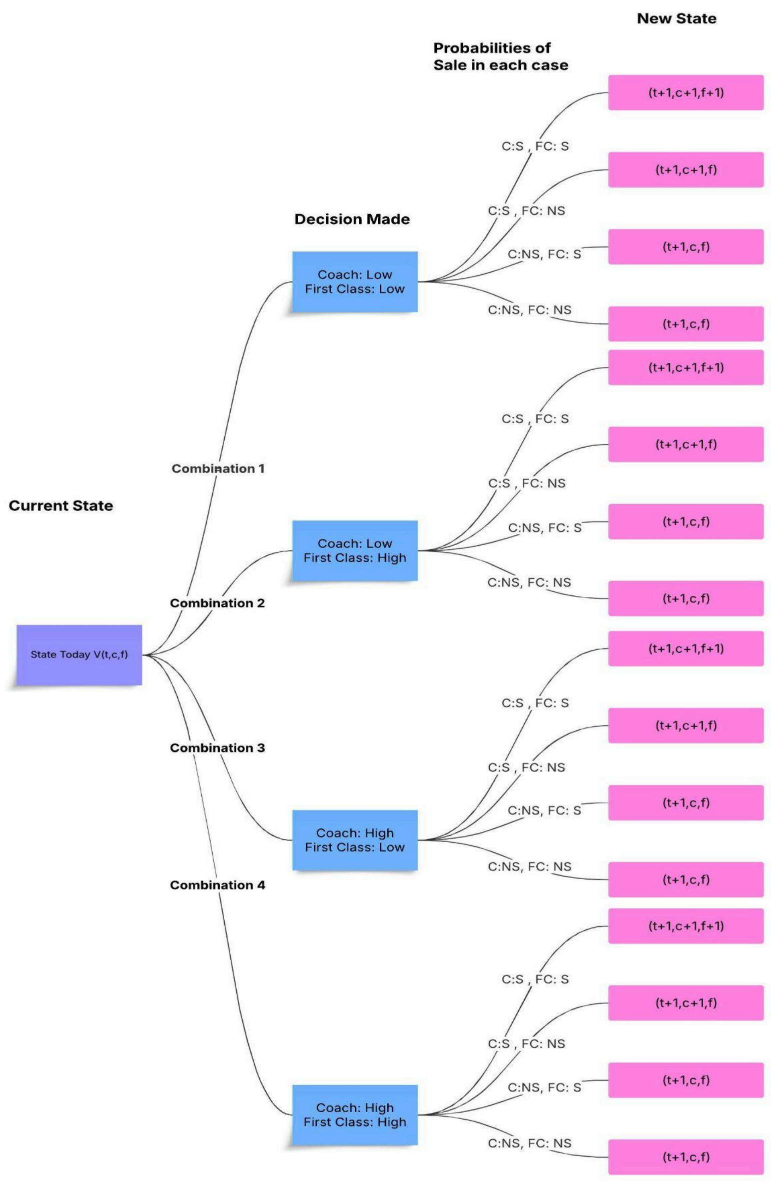
[c' and f' are the number of tickets sold in coach and first class respectively]

There are four possible actions and each of them would lead to a new state as described below:

Outcome	**Coach Sale (*sc*=1/0)**	**First-Class Sale (*sf*=1/0)**	**Transition Probability**	**New State (*t*+1,*c′*,*f′*)**
A	No coach sale (*sc*=0)	No first-class sale (*sf*=0)	(1−prob*c*)(1−prob*f*)	(*t*+1,*c*,*f*)
B	No coach sale (*sc*=0)	First-class sale (*sf*=1)	(1−prob*c*)prob*f*	(*t*+1,*c*,*f*+1)
C	Coach sale (*sc*=1)	No first-class sale (*sf*=0)	prob*c*(1−prob*f*)	(*t*+1,*c*+1,*f*)
D	Coach sale (*sc*=1)	First-class sale (*sf*=1)	prob*c**prob*f*	(*t*+1,*c*+1,*f*+1)

Note that:

- probc and probf depends on the price that we pick for the coach and first class seats
- For the simple case: we do not have the option to not sell the tickets at any time - we need to pick a price for both the tiers
- Because these two events, probability of sale in coach and that in the first class, are two independent events, we could directly multiple these probabilities in each of these scenarios to get the final probability for that scenario



Value Function

$V(t, c, f) = R(t, c, f, p_c, p_f) + \gamma * \sum P(s_c, s_f) * V(t+1, c', f')$

The Value today is the sum of revenue today and the discounted value of all the future values (there are no costs incurred today and all the overbooking cos is incurred at the end of booking period)

We use this to develop the Bellman Equation for the problem in the next part

Bellman Equation:

$$V(t,c,f) = \max_{p_c \in \{300,350\} p_f \in \{425,500\}} \left[E\left[\text{Revenue}\left(p_c, p_f\right)\right] + \gamma \cdot E\left[V(t+1, c', f')\right]\right]$$

Breaking down the 2 components of this equation:

a. <u>Immediate Rewards today:</u>

$$E[\text{Revenue}] = \text{prob}_c(p_c, f) \cdot p_c + \text{prob}_f\left(p_f\right) \cdot p_f (\text{Coach and First-class Revenue})$$

Coach Sale Probability

$$\text{prob}_c = \begin{Bmatrix} 0.65 + 0.03 \times 1_{\{f \geq 20\}} & \text{if} p_c = 300 \text{and} c < C_{\max} \\ 0.30 + 0.03 \times 1_{\{f \geq 20\}} & \text{if} p_c = 350 \text{and} c < C_{\max} \\ 0 & \text{otherwise} \end{Bmatrix}$$

First-class Sale Probability

$$\text{prob}_f = \begin{Bmatrix} 0.08 & \text{if} p_f = 425 \text{and} f < 20 \\ 0.04 & \text{if} p_f = 500 \text{and} f < 20 \\ 0 & \text{otherwise} \end{Bmatrix}$$

b. **Expected Future Value:**

$$E[V(t+1,c',f')] = \sum_{s_c \in \{0,1\}} \sum_{s_f \in \{0,1\}} P(s_c,s_f) \cdot V(t+1,c',f')$$

The daily discount factor:

$$\gamma = \frac{1}{1+0.17/365} \text{(Daily discount factor)}$$

With transition probabilities:

$$P(s_c,s_f) = \begin{Bmatrix} (1-\text{prob}_c)(1-\text{prob}_f) & s_c = 0, s_f = 0 \\ (1-\text{prob}_c) \cdot (\text{prob}_f) & s_c = 0, s_f = 1 \\ (\text{prob}_c) \cdot (1-\text{prob}_f) & s_c = 1, s_f = 0 \\ (\text{prob}_c) \cdot (\text{prob}_f) & s_c = 1, s_f = 1 \end{Bmatrix}$$

and next states as the following:

$$c' = \min(c+sc, C\max), f' = \min(f+sf, 20)$$

Terminal Conditions

The terminal condition in case of overbooking the flights is not 0. We would not get any revenue but would incur costs for either bumping the overbooked passengers in the coach to the first class or to deboard them, in case of no seats available. We incur $50 per customer bumped to the first class from coach and $425 per customer to deboard them.

$$V(0,c,f) = -\sum_{k=0}^{c} \sum_{m=0}^{f} \binom{c}{k} (0.95)^k (0.05)^{c-k} \binom{f}{m} (0.97)^m (0.03)^{f-m} \cdot \text{Cost}(k,m)$$

Components of the terminal conditions:

*It includes the following three components multiplied together - Coach showup * first-class showup * cost in that scenario (in the same order as the equation above)*

a. **Binomial probabilities:**

The first two components, coach and first class show up probabilities, are:

1. Coach showup: Probability that k out of c coach passengers show up (95% show-up rate)
2. First Class showup: Probability that m out of f first class passengers show up (97% show-up rate)

b. **Overbooking Costs:**

Cost(k,m) = 50. min(max(k - 100, 0), 20 - m) + 425. max(max(k - 100, 0) - (20 - m), 0)

- k: Number of coach passengers who show up.
- m: Number of first-class passengers who show up.
- max(k−100,0): Excess coach passengers requiring accommodation.
- 20−m: Available first-class seats after accounting for first-class show-ups.

Example Calculation

Suppose:

- c=105 coach tickets sold (5 overbooked).
- f=18 first-class tickets sold.
- k=103 coach passengers show up.
- m=18 first-class passengers show up.

Overbooking Cost:

- Excess coach passengers: 103−100=3.
- Available first-class seats: 20−18=2.
- Bump 2 passengers to first-class: 2×50=$100
- Deny boarding to 1 passenger: 1×425=$425
- Total cost: $100+$425=$525

Analysis
We start by importing all the necessary packages and initializing the parameters in the question.

```
TIME_STEPS = 365
OVERBOOK_COACH = 5
N_SEATS_COACH_ORIGINAL = 100
N_SEATS_COACH = N_SEATS_COACH_ORIGINAL + OVERBOOK_COACH
N_SEATS_FIRST_CLASS = 20
SHOWUP_PROB_COACH = 0.95
SHOWUP_PROB_FIRST_CLASS = 0.97
PRICE_COACH = [300, 350]
PRICE_FIRST_CLASS = [425, 500]
COACH_PRICE_INCREASE_IF_FC_SOLD_OUT = 0.03
PROB_COACH = [0.65, 0.30]
PROB_FIRST_CLASS = [0.08, 0.04]
COST_UPGRADE = 50
COST_OFF = 425
DISCOUNT = 1/(1+0.17/365)
```

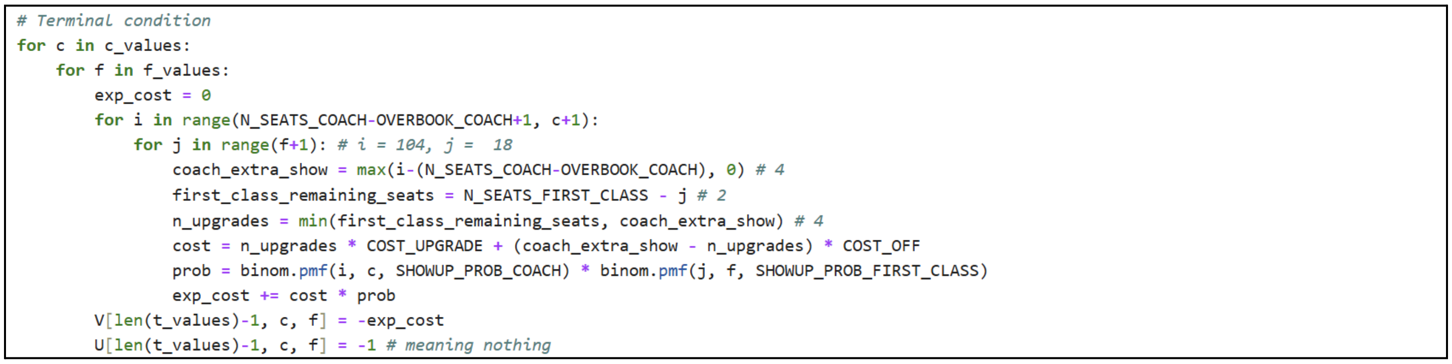
Current baseline: No overbooking:
Before working out the overbooking problem, it would be interesting to know the baseline profit that we would get if there's no overbooking. This is done by putting allowed overbooking of seats to be 0 in the code we would discuss in the next section. We would end up with an expected maximum profit of ***$40,654.29***
This becomes our baseline for all other analysis going forward.
Allowing 5 seats to be overbooked in coach:

```
def policy(max_overbooked_seats, has_option_to_not_sell, apply_seasonality):
    OVERBOOK_COACH = max_overbooked_seats
    N_SEATS_COACH = N_SEATS_COACH_ORIGINAL + OVERBOOK_COACH
    t_values = range(TIME_STEPS+1)
    c_values = range(N_SEATS_COACH+1)
    f_values = range(N_SEATS_FIRST_CLASS+1)
    V = np.zeros((TIME_STEPS+1, N_SEATS_COACH+1, N_SEATS_FIRST_CLASS+1))
    U = np.zeros((TIME_STEPS+1, N_SEATS_COACH+1, N_SEATS_FIRST_CLASS+1))
    # Terminal condition
    for c in c_values:
        for f in f_values:
            exp_cost = 0
            for i in range(N_SEATS_COACH-OVERBOOK_COACH+1, c+1):
                for j in range(f+1): # i = 104, j =  18
                    coach_extra_show = max(i-(N_SEATS_COACH-OVERBOOK_COACH), 0) # 4
                    first_class_remaining_seats = N_SEATS_FIRST_CLASS - j # 2
                    n_upgrades = min(first_class_remaining_seats, coach_extra_show) # 4
                    cost = n_upgrades * COST_UPGRADE + (coach_extra_show - n_upgrades) * COST_OFF
                    prob = binom.pmf(i, c, SHOWUP_PROB_COACH) * binom.pmf(j, f, SHOWUP_PROB_FIRST_CLASS)
                    exp_cost += cost * prob
            V[len(t_values)-1, c, f] = -exp_cost
            U[len(t_values)-1, c, f] = -1 # meaning nothing
    for t in reversed(range(TIME_STEPS)):
        for c_seats in range(N_SEATS_COACH+1):
            for f_seats in range(N_SEATS_FIRST_CLASS+1):
                profit_today = []
                for c in range(2):  # 0 is low price, 1 is high price for coach
                    for f in range(2):  # 0 is low price, 1 is high price for first class
                        expected_profit = 0
                        for sc in range(2):  # coach sale (0 or 1)
                            for sf in range(2):  # first-class sale (0 or 1)
                                prob_c = PROB_COACH[c]
                                if f_seats == N_SEATS_FIRST_CLASS:
                                    prob_c += COACH_PRICE_INCREASE_IF_FC_SOLD_OUT
                                prob_f = PROB_FIRST_CLASS[f]
                                if c_seats == N_SEATS_COACH:
                                    prob_c = 0
                                if f_seats == N_SEATS_FIRST_CLASS:
                                    prob_f = 0
                                pc = prob_c if sc else 1 - prob_c
                                pf = prob_f if sf else 1 - prob_f
                                prob = pc * pf
                                next_c = min(c_seats + sc, N_SEATS_COACH)
                                next_f = min(f_seats + sf, N_SEATS_FIRST_CLASS)
                                reward = sc * PRICE_COACH[c] + sf * PRICE_FIRST_CLASS[f]
                                future_val = V[t + 1, next_c, next_f]

                                expected_profit += prob * (reward + DISCOUNT * future_val)

                        profit_today.append(expected_profit)
                V[t, c_seats, f_seats] = max(profit_today)
                U[t, c_seats, f_seats] = np.argmax(profit_today)

    return V, U
```

Analysing the code here:

The U and V matrices are three dimensional matrices of the dimensions of allowed days+1, available coach seats (including overbooking) +1 and available first class seats+1.
For any given time, we can look at the best possible decision for a given number of seats booked in each of the classes of seats.

We start with the terminal condition:



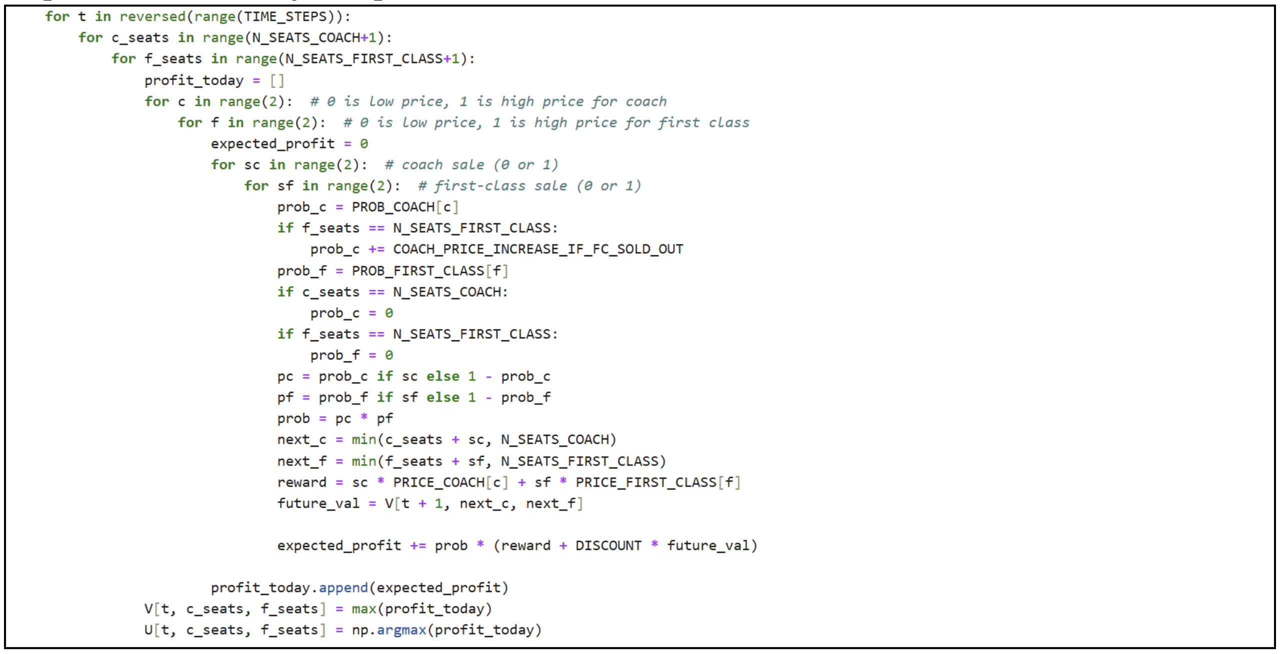
```
# Terminal condition
for c in c_values:
    for f in f_values:
        exp_cost = 0
        for i in range(N_SEATS_COACH-OVERBOOK_COACH+1, c+1):
            for j in range(f+1): # i = 104, j =  18
                coach_extra_show = max(i-(N_SEATS_COACH-OVERBOOK_COACH), 0) # 4
                first_class_remaining_seats = N_SEATS_FIRST_CLASS - j # 2
                n_upgrades = min(first_class_remaining_seats, coach_extra_show) # 4
                cost = n_upgrades * COST_UPGRADE + (coach_extra_show - n_upgrades) * COST_OFF
                prob = binom.pmf(i, c, SHOWUP_PROB_COACH) * binom.pmf(j, f, SHOWUP_PROB_FIRST_CLASS)
                exp_cost += cost * prob
        V[len(t_values)-1, c, f] = -exp_cost
        U[len(t_values)-1, c, f] = -1 # meaning nothing
```

The terminal condition in this case is not zero but the expected costs for either bumping the overbooking passengers to first class or deboarding them.

- For each combination of remaining coach seats (c) and first-class seats (f):
    - The code iterates over possible numbers of passengers showing up in coach (i) and first class (j)
    - Calculates:
        - coach_extra_show: Number of coach passengers who cannot be seated
        - n_upgrades: Number of coach passengers upgraded to first class (if seats are available)
        - cost: Total cost due to upgrades and overbooked penalties
        - prob: Probability of this scenario occurring using the binomial probability mass function (binom.pmf)
    - Accumulates the expected cost (exp_cost) weighted by probabilities.
- Stores the negative expected cost in V (since it's a cost/ penalty)

Once the terminal condition is in place we start from the last but one time period and loop backwards for all the expected values of seats sold.

Loop backwards in time for all possible scenarios:



```
    for t in reversed(range(TIME_STEPS)):
        for c_seats in range(N_SEATS_COACH+1):
            for f_seats in range(N_SEATS_FIRST_CLASS+1):
                profit_today = []
                for c in range(2):  # 0 is low price, 1 is high price for coach
                    for f in range(2):  # 0 is low price, 1 is high price for first class
                        expected_profit = 0
                        for sc in range(2):  # coach sale (0 or 1)
                            for sf in range(2):  # first-class sale (0 or 1)
                                prob_c = PROB_COACH[c]
                                if f_seats == N_SEATS_FIRST_CLASS:
                                    prob_c += COACH_PRICE_INCREASE_IF_FC_SOLD_OUT
                                prob_f = PROB_FIRST_CLASS[f]
                                if c_seats == N_SEATS_COACH:
                                    prob_c = 0
                                if f_seats == N_SEATS_FIRST_CLASS:
                                    prob_f = 0
                                pc = prob_c if sc else 1 - prob_c
                                pf = prob_f if sf else 1 - prob_f
                                prob = pc * pf
                                next_c = min(c_seats + sc, N_SEATS_COACH)
                                next_f = min(f_seats + sf, N_SEATS_FIRST_CLASS)
                                reward = sc * PRICE_COACH[c] + sf * PRICE_FIRST_CLASS[f]
                                future_val = V[t + 1, next_c, next_f]

                                expected_profit += prob * (reward + DISCOUNT * future_val)

                        profit_today.append(expected_profit)
                V[t, c_seats, f_seats] = max(profit_today)
                U[t, c_seats, f_seats] = np.argmax(profit_today)
```

We have 4 different combinations of prices possible for the coach and first class and we loop through each one of them.
For each combination of pricing strategies:

- Iterate over possible ticket sales (sc for coach, sf for first class: 0 or 1)
- Compute probabilities of selling tickets based on current prices and seat availability:

```
prob_c = PROB_COACH[c]
if f_seats == N_SEATS_FIRST_CLASS:
    prob_c += COACH_PRICE_INCREASE_IF_FC_SOLD_OUT
prob_f = PROB_FIRST_CLASS[f]
```

- As mentioned in the question, the probability of selling a coach seat increases by 3% if all the seats in the first class are sold out.
- Another important point to note would be to change the probability of selling a ticket to 0 if all the seats (including the allowed overbooked capacity of seats) are sold out in a particular seat class - if there are no seats left to be sold, there's no probability of us selling anything!
- For each combination of sale and no sale for coach and first class, Compute:
    - Reward from ticket sales (reward)
    - Future value from the next state (future_val), retrieved from V[t + 1]
    - Total expected profit weighted by probabilities

```
pc = prob_c if sc else 1 - prob_c
pf = prob_f if sf else 1 - prob_f
prob = pc * pf
next_c = min(c_seats + sc, N_SEATS_COACH)
next_f = min(f_seats + sf, N_SEATS_FIRST_CLASS)
reward = sc * PRICE_COACH[c] + sf * PRICE_FIRST_CLASS[f]
future_val = V[t + 1, next_c, next_f]

expected_profit += prob * (reward + DISCOUNT * future_val)
```

Because the event of selling a ticket in the coach is independent of selling one in the first class, we can directly multiply the individual probabilities to get the joint probability of those events happening together and use them as weights for the scenario. The final part of this is storing the maximum expected profit values of all states and the optimal decisions to take at each possible state and time in V and U matrices respectively

```
        profit_today.append(expected_profit)
V[t, c_seats, f_seats] = max(profit_today)
U[t, c_seats, f_seats] = np.argmax(profit_today)
```

We then call this function for 5 overbooked coach seats:

```
V, U = policy(max_overbooked_seats=5, has_option_to_not_sell=False, apply_seasonality=False)
V[0][0][0]
```

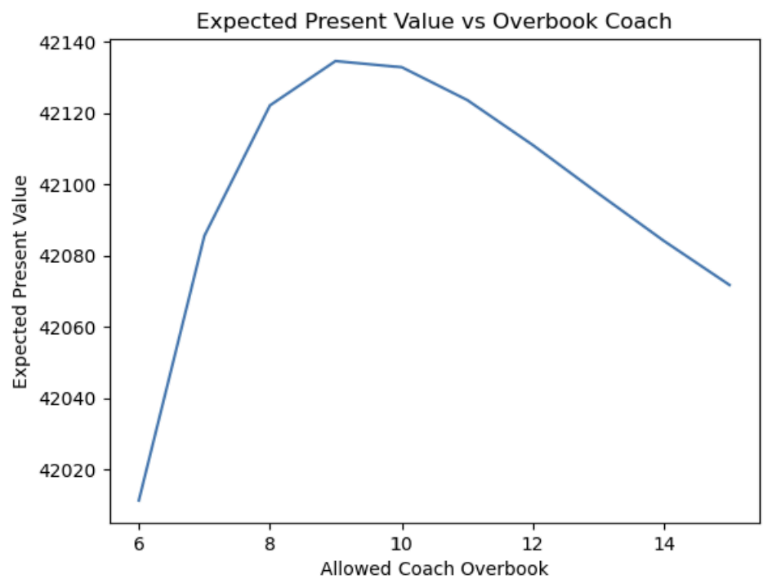
The maximum expected profit using this policy would be : ***$41886.158,*** which is ***3.03%*** higher than allowing no overbooking

Exploring overbooking options from 6 to 15:

To get to the optimal number of overbooking of coach seats allowed, we loop from 6 to 15:

```
V_dict = {}
for OVERBOOK_COACH in range(6, 16):
    V, U = policy(max_overbooked_seats=OVERBOOK_COACH, has_option_to_not_sell=False, apply_seasonality=False)
    V_dict[OVERBOOK_COACH] = V[0, 0, 0]
```

Overbooking Allowed	**Maximum Expected Profit ($)**
6	$42,011.22
7	$42,085.54
8	$42,122.17
9	$42,134.62
10	$42,132.90
11	$42,123.67
12	$42,111.03
13	$42,097.42
14	$42,084.11
15	$42,071.74



At 9 overbooked seats, we get the maximum expected profit of ***$42,134.62,*** which is ***3.64%*** higher than the baseline when there's no overbooking allowed

Allowing for a no-sale option:

If the policy changes to allow days where we can choose not to sell any coach tickets, the following changes are observed in the components of the DP:

1. Decision Variables

- Original Coach: Coach has 2 choices: $300 or $350
- New Policy for coach: Coach has 3 choices: $300, $350, or no sale
- First-class: Remains 2 choices: $425 or $500

Effect: The action space expands from $(pc,pf) \in \{300,350\}\times\{425,500\}$ to $(pc,pf)\in\{300,350,\text{No Sale}\}\times\{425,500\}$

2. Coach Sale Probability

- Original:

$$\text{prob}_c = \begin{cases} 0.65 + 0.03 \times 1_{\{f\geq 20\}} & \text{if} p_c = 300 \text{and} c < C_{\max} \\ 0.30 + 0.03 \times 1_{\{f\geq 20\}} & \text{if} p_c = 350 \text{and} c < C_{\max} \\ 0 & \text{otherwise} \end{cases}$$

- New Policy:

$$\text{prob}_c(p_c, f) = \begin{cases} 0.65 + 0.03 \cdot \mathbb{1}_{\{f\geq 20\}} & \text{if } p_c = 300, \\ 0.30 + 0.03 \cdot \mathbb{1}_{\{f\geq 20\}} & \text{if } p_c = 350, \\ 0 & \text{if } p_c = \text{No Sale or } c \geq C_{\max}. \end{cases}$$

Effect: The airline can now *strategically block coach sales* to preserve overbooking flexibility.

3. Bellman Equation:

- Original: Maximize over 2 coach prices.
- New Policy: Maximize over 3 coach options (including "No Sale"):

$$V(t,c,f) = \max_{p_c\in\{300,350,\text{No Sale}\},p_f\in\{425,500\}} \left[\mathbb{E}[\text{Revenue}] + \gamma \cdot \mathbb{E}[V(t+1,c',f')]\right]$$

where the immediate revenue calculation becomes:

$$\mathbb{E}[\text{Revenue}] = \begin{cases} \text{prob}_c \cdot p_c + \text{prob}_f \cdot p_f & \text{if } p_c \neq \text{No Sale}, \\ \text{prob}_f \cdot p_f & \text{if } p_c = \text{No Sale}. \end{cases}$$

Effect: The airline can trade off immediate revenue for reduced overbooking risk

4. State Transitions

- Original: Coach tickets sold $c'=\min(c+sc,C\max)$
- New Policy:

$$c' = \begin{cases} \min(c + s_c, C_{\max}) & \text{if } p_c \neq \text{No Sale}, \\ c & \text{if } p_c = \text{No Sale}. \end{cases}$$

Effect: "No Sale" days freeze coach ticket sales, preserving capacity for future days
This adds on 2 more possible outcomes

Outcome	**Coach Sale (*sc*=1/0)**	**First-Class Sale (*sf*=1/0)**	**Transition Probability**	**New State (*t*+1,*c*′,*f*′)**
E	No coach sale (*sc*=NO sale)	No first-class sale (*sf*=0)	(1−prob*f*)	(*t*+1,*c*,*f*)
F	No coach sale (*sc*=NO Sale)	First-class sale (*sf*=1)	prob*f*	(*t*+1,*c*,*f*+1)

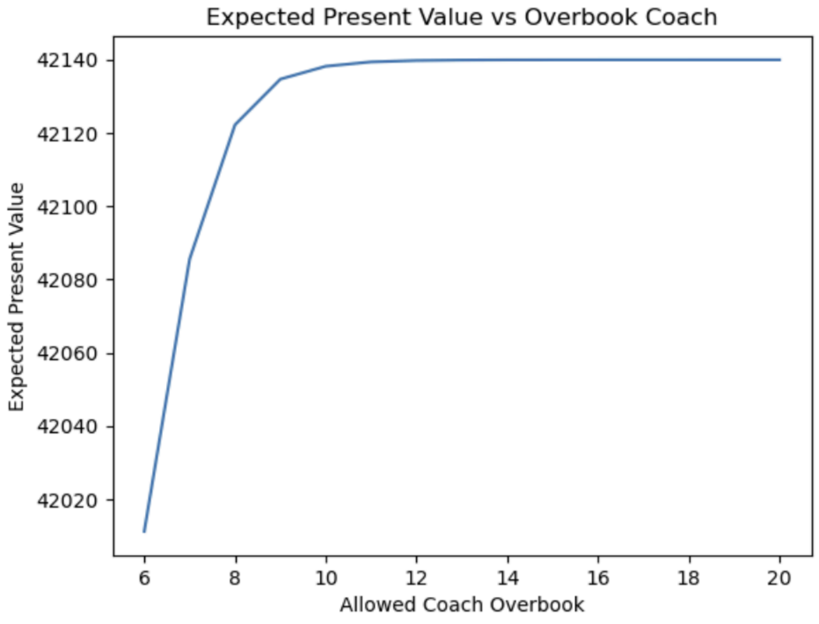
Note that:

- probc and probf still depends on the price that we pick for the coach and first class seats

Everything else remains the same
To implement this, we add the following code to the previous function:

```
if has_option_to_not_sell:
    # add choice to not sell coach
    for f in range(2):  # 0 is low price, 1 is high price for first class
        expected_profit = 0
        for sf in range(2):
            prob_f = PROB_FIRST_CLASS[f]

            if f_seats == N_SEATS_FIRST_CLASS:
                prob_f = 0

            pf = prob_f if sf else 1 - prob_f
            prob = pf

            next_f = min(f_seats + sf, N_SEATS_FIRST_CLASS)
            reward = sf * PRICE_FIRST_CLASS[f]
            future_val = V[t + 1, c_seats, next_f]

            expected_profit += prob * (reward + DISCOUNT * future_val)
        profit_today.append(expected_profit)
```

The parameter has_option_not_to_sell determines whether the airline has the option to stop selling coach tickets entirely at any given time step.
When has_option_to_not_sell is enabled, an extra loop is added to calculate profits for scenarios where no coach tickets are sold. This is done by iterating over first-class ticket sales only and computing their expected profit. The results are then stored in a similar way as in normal overbooking without the no sell policy.
With 20 overbooked seats, the expected maximum profit is ***$42139.8927, 3.65%*** higher than the baseline profit.
Comparing this for different values of overbooking allowed :

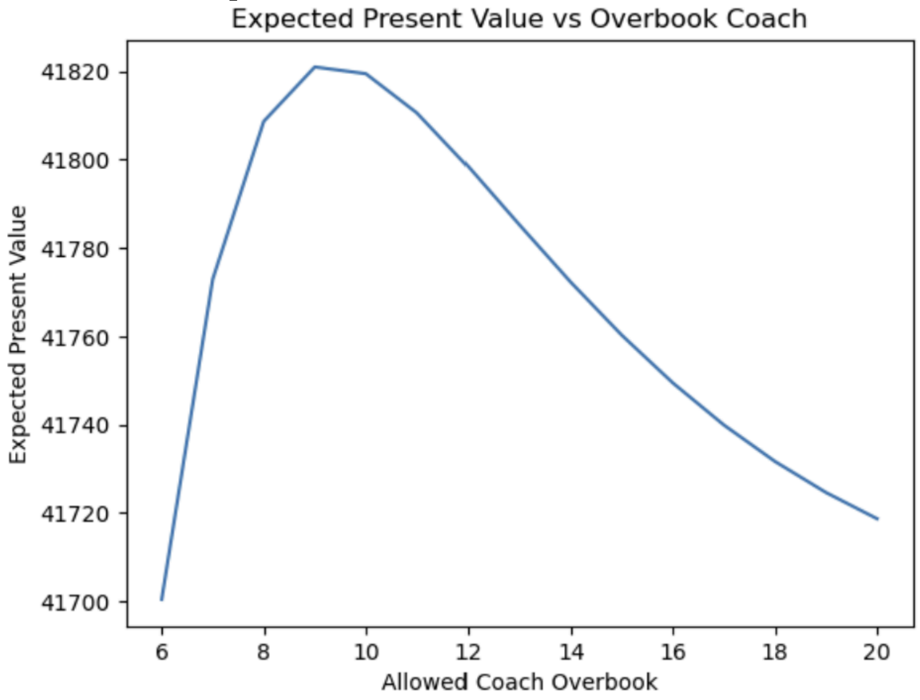


It is observed that the value of the expected maximum profit plateaus after 11 overbooked seats

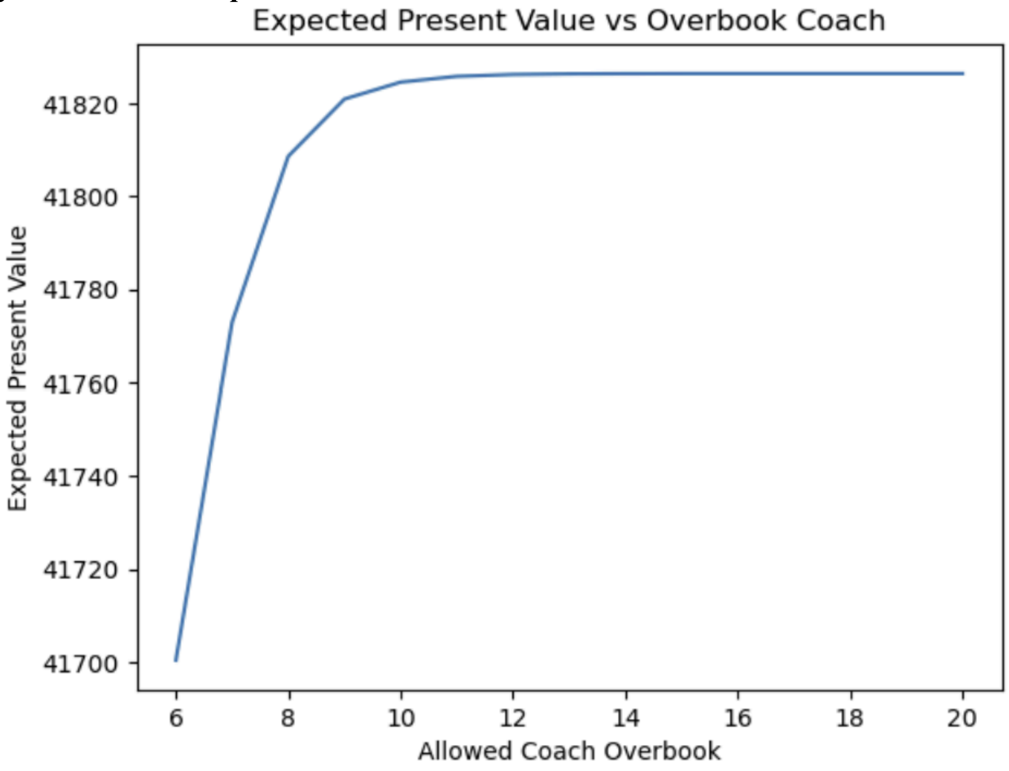
Adding Seasonality
In reality, the probability of sale is not the same for all the days and as we get closer to the take off, there's more probability to sell a ticket. To model the changing demand as we get closer to the flight take off, we make a simple modification in the sale probabilities for coach as well as the first class seats

```
prob_c *= (0.75 + t/730)
prob_f *= (0.75 + t/730)
```

Adding seasonality without the no-sale option:



Maximum value of the expected profit is ***$41820.96 at 9*** overbooked seats allowed. This is just ***2.87%*** higher than the baseline
Adding seasonality with the no sale option:



Maximum value of the expected profit is ***$41826.445 with more than 11*** overbooked seats allowed. This is just ***2.89%*** higher than the baseline

Important Observation:
Adding seasonality to the model results in lower expected profits compared to a similar policy without seasonality. While demand increases closer to the departure date, the early days of the sales window experience weaker demand, which delays revenue generation. Because of the discount factor applied to future profits, selling later becomes less valuable, even if demand is higher. This leads to a drop in overall expected profit.

Forward Simulation of Policies
During each simulation, the most optimal decision at any given time (i.e., the price of coach and first-class tickets for that day) was taken from the U matrix. The probabilities of selling tickets in each class were scaled by the seasonality factor.

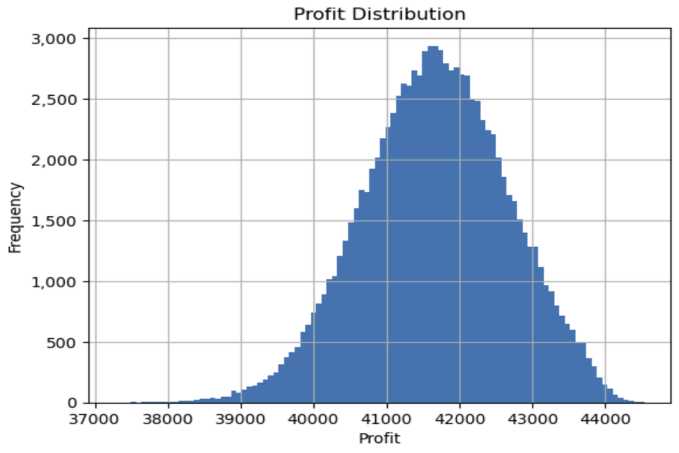
For each day in the simulation:
- Random numbers were used to simulate ticket sales based on the calculated probabilities.
- Ticket counts (c_sold, f_sold) were updated accordingly.
- The daily profit, primarily the revenue from selling that ticket, was calculated and added to the overall profit for that simulation.

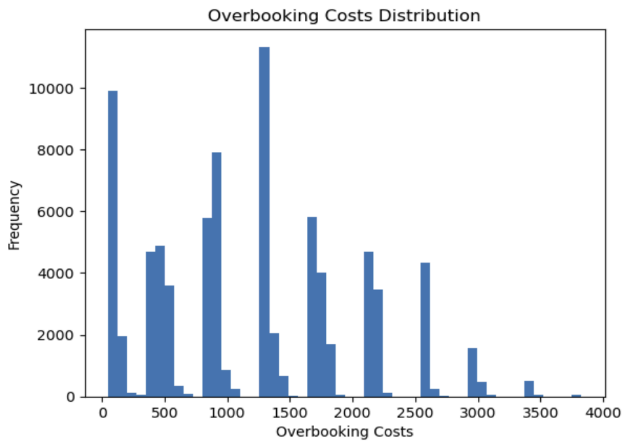
On the day of flight take-off, we generated random numbers to determine the number of passengers who showed up for each class, using the probability distribution and the number of tickets sold (including overbooked tickets).
If more coach passengers showed up than available seats:
- We calculated the upgrade costs (COST_UPGRADE) for moving overbooked coach passengers to first class.
- If first-class was also fully booked, we calculated penalties for deboarded passengers (COST_OFF).
- We updated the counters for overbooking events and appended the costs to the tracking lists.

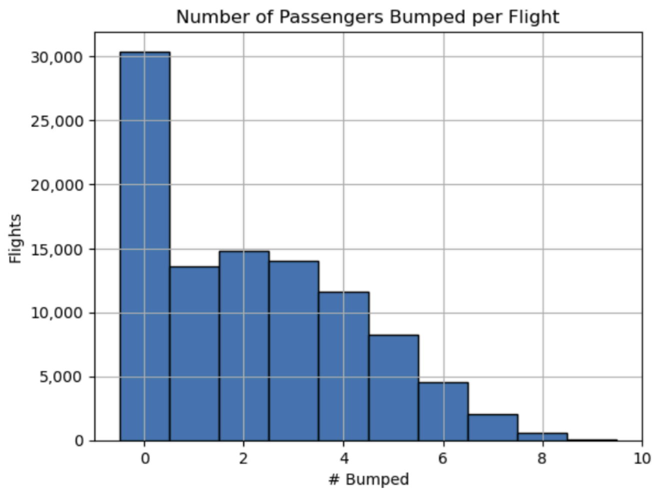
Metrics Analyzed for Each Policy:
- **Expected Profit and Standard Deviation:** Calculated as the mean and standard deviation of the profit_lst values.
- **Overbooking Analysis:**
  - Percentage of times the flight was overbooked given by overbook_count/number_of_simulations* 100
  - Percentage of times when we had to de-board passengers because we had no seats left in either of the coaches. This is given by: kick_count/number_of_simualtions*100
- **Overbooking Costs:**
  - Average overbooking costs, including both upgrade and de-boarding penalties.
  - Standard deviation of the average costs.
- **Pricing Decisions and Blocked Upgrades:**
  - Percentage of flights where the first-class section was full, preventing upgrades.

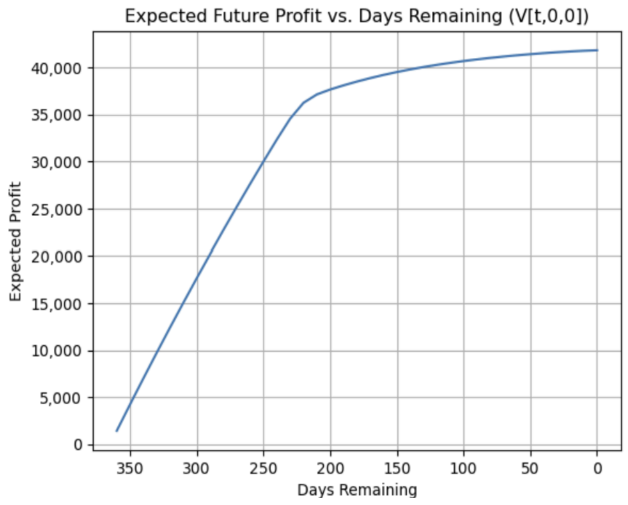
Case 1 : Seasonality incorporated, without the option of no-sell : allowing 9 seats to be overbooked was used because that gave us the maximum expected profits :
- Dynamic Programming Value: $41,825.07
- Expected Profit: $41,673.78
- Standard Deviation: $990.21
- Overbooked Flights: 81.64%
- Flights with At Least One Kicked Passenger: 69.64%
- Average Overbooking Costs: $1,186.65
- Standard Deviation of Costs: $804.24
- Pricing Strategy:
  - Coach 'Low' used 4.42%
  - Coach 'High' used 95.58%

    - No-Sell Coach used 0.00%
- Blocked Upgrades: 32.34%



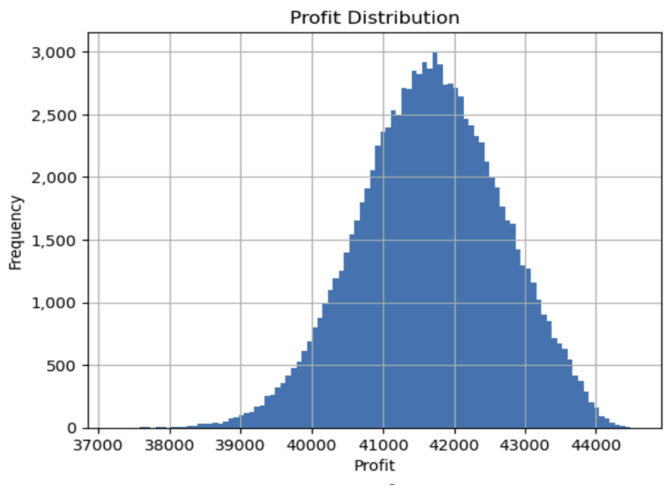


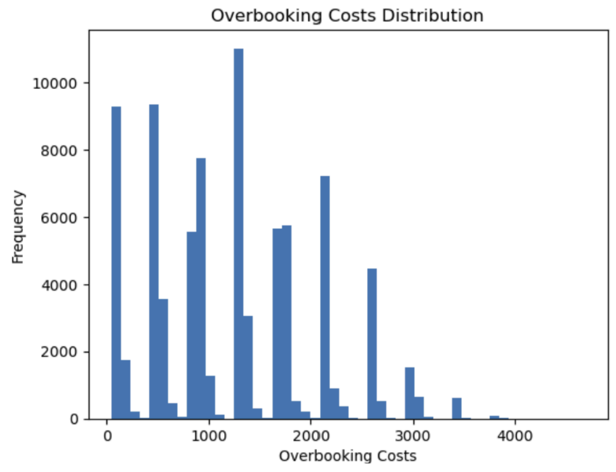


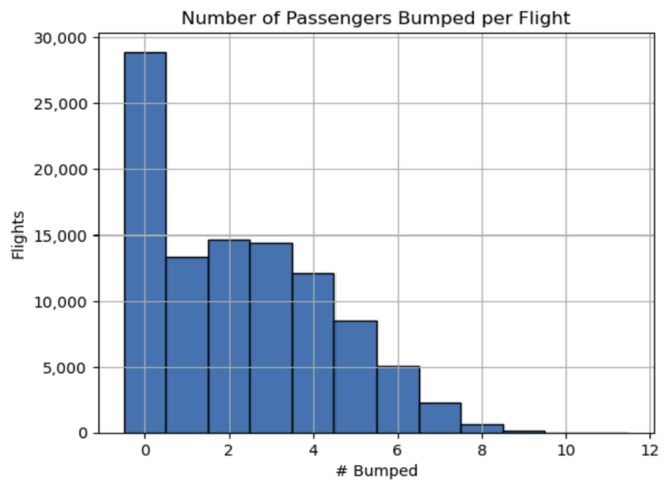


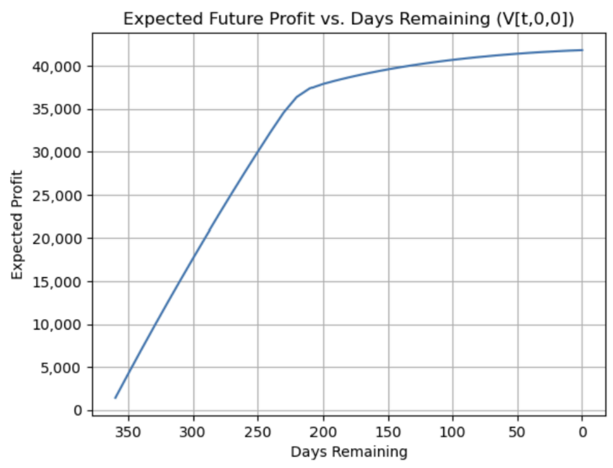
Case 2 : Seasonality incorporated, with the option of no-sell: allowing 20 seats to be overbooked was used because that gave us the maximum expected profits
(the profit was almost similar for anything beyond 11 overbooked seats allowed but using 20 gives the best possible scenario) ***:***

- Dynamic Programming Value: $41,830.27
- Expected Profit: $41,668.17
- Standard Deviation: $999.81
- Overbooked Flights: 82.39%
- Flights with At Least One Kicked Passenger: 70.75%
- Average Overbooking Costs: $1,231.08
- Standard Deviation of Costs: $811.44
- Pricing Strategy:
    - Coach 'Low' used 1.55%
    - Coach 'High' used 98.45%
    - No-Sell Coach used 2.80%
- Blocked Upgrades: 31.36%









Observations:

- In both scenarios, the profit distribution follows a normal pattern centered around $41,000. Most flights experienced moderate overbooking costs, with occasional high-cost outliers resulting from significant overbooking or passenger bumping.
- Approximately 65% of the time, 0-2 coach passengers were bumped to first class.
- The choice between incorporating a no-sell option or not has a marginal impact on expected profit. Allowing higher overbooking limits slightly increases the likelihood of passenger bumping but does not significantly affect overall profitability.
- The strategic use of the no-sell option is more beneficial when considering customer satisfaction and the potential negative effects of overbooking.

Results: Comparing Different Policies

- **Result 1**: Profit from allowing 5 seats in coach to be overbooked = ***$41886.158 (3.03% higher than no overbooking)***
- **Result 2**: Allowing overbooking by 9 seats leads to the maximum expected profit of ***$42,134.62 (3.64%*** higher than the baseline, no overbooking policy)
- **Result 3**: Allowing No-Sell Policy for the coach seats : Maximum expected profit = ***$42,139.89,*** by allowing more than 11 seats to be overbooked - maximum at 20 (***3.65%*** higher than the baseline)
- **Result 4**: Adding seasonality:
  - Part a: Without No sell option: Allowing overbooking by 9 seats leads to the maximum expected profit of ***$41820.96 (2.87%*** higher than the baseline, no overbooking policy)
  - Part b: With the No-sell option: Maximum expected profit = ***$41826.445,*** by allowing more than 11 seats to be overbooked - maximum at 20 (***2.89%*** higher than the baseline)

Comparing the results from different policies

Policy	**Overbooking allowed (seats)**	**Profit (in $)**	**Improvement from baseline (compared to 0 overbooking)**
No seasonality, without no sell option	9	$42,134.62	3.64%
No seasonality, with no sell option	12-20	$42,139.89	3.65%
Seasonality without no sell options	9	$41,820.96	2.87%
Seasonality with no sell option	12-20	$41,826.45	2.89%

*highlight indicates highest level of profit, but due to a more realistic approach the most favorable policy is highlighted

Final Recommendation: After testing a wide range of scenarios, we recommend allowing up to twenty coach seats to be overbooked and including the no sell option for coach ticket sales. We chose to include seasonality in our model since it better reflects how customer demand tends to increase as the flight date approaches. With this setup, the expected discounted profit reaches $41,826.45, the highest among all strategies we evaluated.

Conclusion

This project demonstrates the power and versatility of dynamic programming in optimizing airline revenue through strategic overbooking and ticket pricing. By modeling the complex decision-making process involving ticket sales, overbooking costs, and customer behavior, we have developed a robust framework that maximizes expected discounted profits for a hypothetical airline operating over a one-year period.

Our analysis highlights the importance of carefully balancing overbooking levels to maximize profit while minimizing customer dissatisfaction. The results clearly indicate that moderate overbooking (up to nine seats) is optimal, as it strikes a balance between additional revenue from overbooked seats and the potential costs associated with customer compensation or upgrades. Incorporating seasonality and the option to halt ticket sales provides further insight into how airlines can better manage varying demand patterns and mitigate financial risks.

The forward simulation results reinforce the dynamic programming outcomes, showing consistency in profit margins and overbooking costs across different policy scenarios. Additionally, the model's ability to account for price sensitivity and seasonality provides a more realistic approach to revenue management. As demonstrated, seasonality can slightly decrease profit margins, as the uncertainty of early ticket sales impacts revenue despite increased demand closer to departure.

By systematically comparing different overbooking policies, we have demonstrated that flexible pricing strategies and controlled overbooking can significantly improve financial outcomes. The project underscores the critical role of data-driven decision-making in the aviation industry, where optimizing each revenue stream can have a substantial cumulative impact on overall profitability.

Future Scope

This project lays the foundation for more advanced revenue management models by introducing dynamic pricing and overbooking optimization. However, several potential extensions and improvements could further enhance the model's accuracy and applicability:

1. Incorporating Multi-Segment Flights: Extending the model to consider connecting flights and multi-leg journeys, where overbooking on one segment can impact subsequent legs.
2. Competitor Pricing Analysis: Including dynamic competitor pricing would allow the model to adapt to market conditions and enhance profitability predictions.
3. Granular Customer Behavior Modeling: Integrating data on passenger demographics and booking patterns could improve the accuracy of demand predictions and price sensitivity estimates.
4. Real-Time Data Integration: Leveraging live booking data would make the model more responsive to sudden changes in demand, such as those caused by weather disruptions or significant events.
5. Advanced Machine Learning Techniques: Incorporating reinforcement learning or deep Q-learning could refine the decision-making process by dynamically adjusting strategies based on real-time data.

6. Evaluating Long-Term Customer Impact: Studying how overbooking and pricing policies affect long-term customer loyalty and brand perception, helping airlines strike a balance between short-term revenue and long-term customer satisfaction.
7. Enhanced Cost Modeling: Including variable compensation costs and the impact of customer dissatisfaction on future ticket sales could make the model more comprehensive.

By implementing these extensions, future studies can provide a more nuanced understanding of revenue optimization in the airline industry, leading to smarter and more adaptive pricing strategies that account for real-world complexities.Future work could explore more dynamic pricing models, integrating competitor analysis, and incorporating more granular passenger behavior data. Additionally, refining the cost structure and considering multiple flight segments could further enhance model accuracy.