RM 294 Optimization II - Project II: Dynamic Programming

Prepared For:

Dr. Daniel Mitchell
McCombs School of Business
The University of Texas at Austin
2110 Speedway, Austin, TX 78705

Prepared by:

Dhruv Arora
Areeba Shah
Advaith Shankar
Ethan Wong

Sunday, April 6, 2025

I. TABLE OF CONTENTS

II. DEFINING THE BUSINESS PROBLEM

Airlines operate in a highly competitive environment where profit margins are thin and demand uncertainty is high. One critical decision in this context is how to price tickets over time and how many tickets to sell for an upcoming flight – even beyond the plane's physical capacity. Overbooking (selling more tickets than seats) can increase revenue by accounting for no-shows, but it also risks incurring significant costs and customer dissatisfaction if too many passengers show up. Our project addresses this trade-off by optimizing a joint **pricing and capacity (overbooking) policy** for a flight with two cabins (coach and first-class) over a 365-day ticket-selling horizon. We consider distinct pricing options and demand patterns for coach vs. first-class, the probability that each ticketed passenger actually shows up for the flight, and the financial penalties of accommodating or bumping excess passengers. The goal is to quantify whether an overbooking strategy can **improve expected profit** and, if so, determine the optimal level of overbooking and the best pricing strategy to maximize that profit over the year.

Business Context: The flight in question has 100 coach seats and 20 first-class seats. Historically, about 95% of coach ticket holders show up for the flight, and about 97% of first-class ticket holders show up. When more passengers arrive than there are seats in coach, the airline has two ways to handle the overflow: (1) bump some coach passengers up to first-class (if there are empty first-class seats) at a cost of $50 each, or (2) if both cabins are full, deny boarding to coach passengers (bump off) at a much steeper cost of $425 each (essentially compensating them for the inconvenience). These **overbooking costs** reflect lost goodwill and compensation expenses. To avoid upsetting premium customers, our manager refuses to overbook first-class at all, but is open to overbooking coach class. Thus, any overbooking policy will apply **only to coach tickets**. We also have two price options for each class that influence daily demand: in coach, we can charge either $300 (lower price) or $350 (higher price), and in first-class $425 or $500, with lower prices naturally stimulating higher booking probabilities. For example, on a given day with seats available, a $300 coach fare yields a 65% chance of selling a coach ticket, whereas a $350 fare yields only a 30% chance. First-class demand is more rare: a $425 first-class fare has about an 8% daily booking probability versus 4% at $500. Importantly, these probabilities are interdependent: if first-class is sold out, some would-be first-class customers may book coach instead, effectively boosting coach's sale probability by ~3%. We incorporate this by adding 0.03 to the coach sale probability whenever the first-class cabin is full. Our analysis needs to decide, for each day leading up to the flight, what price to set for coach and first-class, and whether to stop selling tickets once a certain number is sold, in order to maximize the **expected discounted profit** (ticket revenue minus expected overbooking costs) over the year.

In summary, the business problem is to find an optimal strategy that answers:
Should we overbook coach seats, and by how many, and how should we price tickets each day given a limited time to sell and uncertain demand?

We approach this problem using a dynamic programming model to make sequential pricing and selling decisions, then evaluate different overbooking policies (from conservative to aggressive) and even consider a more flexible policy that allows deliberately pausing sales on some days. Finally, we simulate these policies to understand their performance and risk, and we provide recommendations on whether overbooking is worthwhile and under what conditions.

III. APPROXIMATING THE OPTIMAL PRICING AND CAPACITY POLICY

Dynamic Programming Framework

To tackle the sequential decision problem of pricing and overbooking, we developed a **dynamic programming (DP)** model. Dynamic programming allows us to break the year-long booking horizon into daily decisions that consider the remaining time and remaining capacity. We define the **state** at any given day by three variables:

- **s_c:** the number of coach tickets sold so far (this can exceed the physical coach capacity if we allow overbooking),
- **s_f:** the number of first-class tickets sold so far (capped at the first-class capacity of 20),
- **t:** the number of days remaining until the flight (with t=0 meaning departure day).

Each day corresponds to a decision point where we choose an **action**: set a price for a coach (either $300 or $350) and a price for first-class (either $425 or $500). Thus, on a normal day the action space has 2 choices for coach × 2 choices for first-class = 4 possible pricing combinations. We did not initially allow an explicit "do nothing" action – both cabins attempt to sell if not already at their limits. However, we will later introduce a new action that allows us to refrain from selling coach tickets on a day if that proves beneficial (the "no-sale" option).

State Space Definition: We discretize the state space such that s_c in [0, M_c + overbook] and s_f in [0, M_f], where M_c=100 and M_f=20 are the physical seat capacities. The value of "overbook" is a parameter representing how many extra coach tickets we are willing to sell beyond 100 (if any). For example, with an overbook allowance of 5, s_c can range up to 105. The time dimension t in [0, T] with T=365 days (we count downwards, so initially t=365 and on departure day t=0). This DP formulation allows the policy to depend on how many tickets have been sold and how much time is left to sell more.

Transition Probabilities & Rewards: On each day in state (s_c, s_f, t), given a choice of prices, there is uncertainty in whether 0, 1, or 2 tickets are sold (at most one coach and at most one first-class ticket can sell in a single day, according to the problem assumptions). The probabilities of each sale outcome depend on the prices chosen and current availability. We encoded the given demand probabilities into our model. For Coach, we wrote a helper function,

coach_prob(price_coach, fc_sold_out, coach_full), that returns the probability of selling a coach ticket on that day. If we price coach at $300, the base probability of a sale is 0.65; at $350, it is 0.30.

```
def coach_prob(price_coach, fc_sold_out, coach_full):
    if coach_full:
        return 0.0
    if price_coach == 300:
        base_p = 0.65
    else:
        base_p = 0.30
    if fc_sold_out:
        base_p += 0.03
    return min(base_p, 1.0)
```

If first-class is sold out **(fc_sold_out=True)**, we boost the coach sale probability by 0.03 (i.e., 68% instead of 65%, or 33% instead of 30%). And if the coach is already full (meaning s_c is at the allowed maximum for that overbooking policy), then **coach_prob** returns 0 – no coach sale can occur because we've hit the limit. We created a similar function, **fc_prob(price_fc, coach_full, fc_sold)** for first-class.

```
def fc_prob(price_fc, coach_full, fc_sold):
    if coach_full:
        return 0.0
    if fc_sold >= fc_max:
        return 0.0
    if price_fc == 425:
        return 0.08
    else:
        return 0.04
```

A first-class ticket sells with probability 0.08 at $425 or 0.04 at $500 unless the first-class ticket is already sold out (then probability 0). One nuance: we assumed if the coach class is at its overbooking limit **(coach_full=True)**, we do **not** sell further first-class tickets either. This effectively means that once we've hit the maximum coach sales allowed, we stop all sales in our model. In practice, that situation would only arise if we sold the absolute maximum coach tickets early; then the optimal strategy is likely to halt additional sales anyway. This assumption simplified the logic by preventing a scenario where we oversell coach and then still try to sell first-class on top of that.

Given these probabilities, the immediate **reward** on a day (if tickets are sold) is the revenue from any sale(s) that occur: $300 or $350 for a coach sale, and $425 or $500 for a first-class sale. The state transitions as follows: if a coach ticket sells, s_c increases by 1 (up to the allowed max); if a

first-class ticket sells, s_f increases by 1 (up to 20). The day count t decreases by 1. If no ticket sells, the state only changes in t. Because at most one of each class can sell per day, there are four possible outcomes for a given action: *(no sale), (coach sale only), (first-class sale only), (both sales)*. We incorporated all of these in the DP recurrence.

Terminal Condition Policy: At the end of the selling horizon (t = 0), no further sales are allowed. Instead, we compute the **expected overbooking cost** for the tickets already sold. This forms the terminal condition of our DP. The value function is initialized as:V(s_c, s_f, t = 0) = − E[Overbooking Cost | s_c, s_f].To compute this, we wrote the `overbooking_cost(sold_coach, sold_first)` function, which calculates the expected monetary penalty from bumping excess passengers. Using binomial distributions, we simulate show-ups for coach and first-class (with p = 0.95 and 0.97 respectively), then compute the overflow from coach (if any) and whether that overflow can be absorbed by unused first-class seats (bump-up at $50) or not (bump-off at $425). These expected costs are then **subtracted from profit** to initialize the terminal value function.

Bellman Equation: We solve this problem by backward induction. On the **final day (departure day, t=0)**, no further sales can occur – instead, we incur any overbooking cost if we have sold too many tickets. We initialize the value function V(s_c, s_f, t=0) as the **negative** of the expected overbooking cost for having sold s_c coach and s_f first-class tickets when the flight is about to depart. This expected cost is computed using the binomial distribution of show-ups. We wrote an **overbooking_cost(sold_coach, sold_first)** function that calculates the expected total bump cost on departure, given how many tickets were sold.

```
def overbooking_cost(sold_coach, sold_first,
                     M_c=M_c, M_f=M_f,
                     p_show_c=p_show_c, p_show_f=p_show_f,
                     cost_bump_up=cost_bump_up, cost_bump_off=cost_bump_off,
                     fc_max=M_f):
    expected_cost = 0.0
    for k in range(sold_coach + 1):
        # Probability exactly k coach passengers show up
        prob_k = binom.pmf(k, sold_coach, p_show_c)
        for m in range(sold_first + 1):
            # Probability exactly m first-class passengers show up
            prob_m = binom.pmf(m, sold_first, p_show_f)
            # Overflow in coach: passengers above the physical capacity
            overflow_coach = max(k - M_c, 0)
            # Remaining seats in first-class using fc_max
            leftover_first = max(fc_max - m, 0)
            # We can bump up to leftover_first passengers from the coach overflDow
            bumped_up = min(overflow_coach, leftover_first)
            bumped_off = overflow_coach - bumped_up
            scenario_cost = cost_bump_up * bumped_up + cost_bump_off * bumped_off
            expected_cost += scenario_cost * prob_k * prob_m
    return expected_cost
```

Essentially, for each possible number of coach show-ups k (out of sold_coach) and first-class show-ups m (out of sold_first), we determine how many coach passengers would need to be bumped. If k > 100 (coach capacity), the overflow is k - 100. Those overflow coach passengers can be bumped up to any of the 20 - m empty first-class seats (if m<20) at $50 each; any remaining overflow beyond that gets bumped off at $425 each. We weight these costs by the probability of each (k,m) outcome (using **scipy.stats.binom.pmf** with p_show=0.95 or 0.97) and sum them up. This gives E[cost | s_c, s_f], and we set V(s_c, s_f, 0) = -E[cost] since costs reduce profit.

From t=0 (departure) back to t=365 (start), we recursively compute the optimal value and policy. On any day t>0, for each state (s_c, s_f, t), we evaluate each possible action (price combination). We compute the **expected discounted value** of that action: the immediate expected revenue from that day's sales (based on the sale probabilities) **plus** the expected future value V(s_c', s_f', t-1) of the next state, discounted by a daily factor delta = 1/(1+0.17/365) (reflecting a 17% annual discount rate). We chose a daily discount so that revenue earned later is slightly less valuable than revenue earned earlier. The Bellman equation can be summarized as:

$$\mathbf{Q((sc,sf,t),\ a)=E[revenue\_a]+\delta\sum outcomes\ P(outcome \mid a)\ *V(next\ state,t-1)}$$

We then select the action a *that maximizes this Q-value and set V(s_c,s_f,t) = Q((s_c,s_f,t), a)*. In practice, this means that for each state, we check the four possible combinations of coach and first-class prices, compute the expected total profit if we choose that pricing, and record the best one. The code snippet below shows how we iterated over actions and outcomes in our DP:

```python
# Backward recursion for t = T-1 down to 0
for t in reversed(range(T)):
    for s_c in range(max_coach+1):
        for s_f in range(max_first+1):
            best_value = -1e15
            best_coach_choice = 0
            best_first_choice = 0

            # For each possible price choice:
            # Coach: low (300) or high (350)
            # First-class: low (425) or high (500)
            for coach_choice in [0, 1]:
                # Determine if first-class is sold out and if coach is at its limit
                fc_sold_out = (s_f >= fc_max)
                coach_full = (s_c >= max_coach)
                # Get coach sale probability using helper function
                p_c = coach_prob(price_coach[coach_choice], fc_sold_out, coach_full)

                for first_choice in [0, 1]:
                    # Get first-class sale probability using helper function
                    p_f = fc_prob(price_first[first_choice], coach_full, s_f)

                    # Clamp next state indices so that we never exceed our state space
                    next_s_c = min(s_c + 1, max_coach)
                    next_s_f = min(s_f + 1, max_first)

                    # Immediate revenue if a sale occurs:
                    revenue_coach = price_coach[coach_choice]
                    revenue_first = price_first[first_choice]

                    # Compute the expected value over the four outcomes:
                    # (0,0): No sale in either class
                    val_00 = (1 - p_c) * (1 - p_f) * (0.0 + delta * V[s_c, s_f, t+1])
                    # (1,0): Coach sale only
                    val_10 = p_c * (1 - p_f) * (revenue_coach + delta * V[next_s_c, s_f, t+1])
                    # (0,1): First-class sale only
                    val_01 = (1 - p_c) * p_f * (revenue_first + delta * V[s_c, next_s_f, t+1])
                    # (1,1): Both sales occur
                    val_11 = p_c * p_f * ((revenue_coach + revenue_first) + delta * V[next_s_c, next_s_f, t+1])

                    total_val = val_00 + val_10 + val_01 + val_11

                    if total_val > best_value:
                        best_value = total_val
                        best_coach_choice = coach_choice
                        best_first_choice = first_choice

            V[s_c, s_f, t] = best_value
            policy_coach[s_c, s_f, t] = best_coach_choice
            policy_first[s_c, s_f, t] = best_first_choice
```

This loop (implemented in our **solve_airline_overbooking()** function) is executed for every state (s_c,s_f,t) from t=364 down to t=0. The DP state space is fairly large – for example, with an overbooking limit of 10, s_c goes up to 110 and s_f up to 20, yielding 111×21≈2331 states per time step, times 365 days, about 850,000 states total. However, the computation is still tractable in Python, taking on the order of tens of seconds to run, thanks to relatively small action space and our use of efficient NumPy arrays for V.

By the end of this DP, we obtain: (1) a value function V[0,0,365] which is the **maximum expected discounted profit** starting with no tickets sold and 365 days to go, and (2) an optimal

policy map that tells us, for each state, which coach price and first-class price to set. Next, we use this model to evaluate different overbooking policies.

Comparative Analysis of Different Oversell Limits

Our first analysis was to determine whether allowing overbooking improves profit, and if so, by how much and at what optimal level. We started by examining a conservative policy: allowing Coach to be oversold by 5 seats (i.e., up to 105 tickets) as a test case. **By setting the overbooking seat limit to 5, we obtain an expected discounted profit of $41,792.40.** Then we incrementally increased the overbooking limit to 6, 7, …, up to 15 and recorded the results for each scenario. For each case, the DP was solved from scratch with the new state space limit. Below is an excerpt of the output of our program as we varied the overbooking parameter:

Overbooking Limit	Expected Discounted Profit
5	$41,792.40
6	$41,930.97
7	$42,017.84
8	$42,065.71
9	$42,087.94
10	**$42,094.43**
11	$42,091.94
12	$42,084.78
13	$42,075.60
14	$42,065.93
15	$42,056.61

As shown, the **expected profit increases as we allow more overbooking initially**, maximizing at $42,094.43 with a 10-seat overbook. The profit gains actually decline beyond 10 seats. In essence, there is **diminishing return** to overbooking – once we reach 10 extra tickets, we have captured almost all the benefit of hedging against no-shows, and any further sales just create too high a risk of expensive compensation. The difference in expected profit between the 5-overbook scenario and the 10-overbook scenario is about $302, which, while not huge (around 0.7% of total profit), is significant for an airline operating on thin margins. This suggests that a moderate level of overbooking can yield a notable uptick in revenue by better utilizing seats that would otherwise go empty.

The DP policy behaves under different overbooking limits. With a low overbooking allowance (5), the policy tends to become cautious once s_c nears 105 – as we approach the cap, the model

will start favoring the higher coach price (or effectively slowing sales) to avoid hitting the limit too soon. With a higher allowance (like 10), the policy can afford to sell more aggressively (using the low $300 price more often early on) because it has more buffers to accommodate no-shows. However, even with an overbooking cap of 10, the DP does not recklessly sell tickets without bound; it still carefully manages pricing, especially as t gets small or s_c gets close to 110, to balance the last-minute risk.

In summary, our comparative analysis of oversell limits concluded that **allowing coach to be overbooked by up to 10 seats maximizes expected profit**, beating out any lower or higher cap. This provides a data-driven answer to "how much to overbook": around 10 extra tickets for this flight. In practical terms, selling 110 coach tickets for a 100-seat coach cabin is the sweet spot, given the 95% show-up rate.

Introducing a No-Sale Option

After identifying 10 as the optimal overbooking level under the conventional approach, we next explored (Project 2, Part 3) a more flexible strategy: allowing the airline **not to sell** a coach ticket on a day even if there is demand. In other words, on each day, we introduce a third choice for the coach cabin: besides "sell at $300" or "sell at $350," we add "offer no coach seat for sale today." This "no-sale" action could be beneficial if we have already sold a lot of tickets relative to the time remaining – essentially, it gives the model the ability to intentionally hold back sales to avoid overbooking risk rather than relying solely on higher prices to throttle demand. We did **not** introduce a no-sale option for first-class (since first-class wasn't being overbooked and generally has low demand anyway).

To incorporate this, we expanded the coach action space to {0: $300, 1: $350, 2: No Sale}. In our DP code, this meant an extra branch where if **coach_choice == 2**, we set the probability of a coach sale p_c = 0 (no matter the demand).

```
# coach_choice in [0,1,2] => [low price, high price, no sale]
for coach_choice in [0, 1, 2]:

    # Probability of selling a coach ticket
    if coach_choice == 2:
        # "No sale" => p_c = 0
        p_c = 0.0
    else:
        if s_c >= max_coach:
            p_c = 0.0
        else:
            # +3% if first-class is sold out
            if s_f >= max_first:
                p_c = p_coach_day_fullF[coach_choice]
            else:
                p_c = p_coach_day[coach_choice]
```

The first-class decision remained binary. This modification required re-solving the DP because the state space and transition logic changed slightly – we have a new policy dimension to

consider. Intuitively, we expect the no-sale option to be used in scenarios where we are nearing the overbooking limit and still have plenty of time left (so we can afford to skip selling now and perhaps sell later if needed). It offers a dynamic way to avoid overshooting the optimal ticket sales count.

We reran the DP with the no-sale option enabled for Coach (using the same overbooking cap of 10 that we found optimal). The result was a slightly higher expected profit. The snippet below shows the comparison of the optimal profits with and without the no-sale action:

```
Best overbooking (hard cap) profit = 42094.4307402182
Profit with 3-choice (no-sale) option = 42138.14049791319
```

Under the 3-choice policy, the expected discounted profit from the start state increased to $42,138.14, which is an **improvement of $43.71** over the best two-price policy (42094.43). This is a modest gain of about 0.1%, indicating that the no-sale option is used sparingly by the optimal policy – but it *is* used and does confer a small benefit. Essentially, the DP learned to occasionally skip selling a coach seat on certain days when the system was in a high-risk state (e.g., too many tickets sold too far from departure). Strategically, this demonstrates a more nuanced way to control overbooking: instead of a fixed cap alone, the policy can conditionally decide **not** to continue selling even before hitting the hard cap if it's not necessary. It's a form of **risk-aware dynamic throttling** of sales.

From a practical perspective, the no-sale option might correspond to temporarily closing coach sales (perhaps offering only first-class or simply saying coach is sold out for the day) when we've sold a lot of coach tickets ahead of schedule. Our results show this strategy to be slightly more profitable and robust. While the profit gain is small in magnitude, it highlights that **having the flexibility to halt sales** can outperform a rigid policy that always tries to sell until a fixed number. The improvement, though small, is "free money" in the sense that it reduces risk with virtually no downside.

It's worth noting that in our model, the no-sale option was primarily utilized in scenarios where t was still large and s_c was very close to the limit. In those cases, the optimal decision might be to hold off selling more coach tickets for a while (especially if first-class can still be sold) to avoid a high chance of incurring bump costs later. If demand remains strong, the airline can always resume selling (perhaps at a higher price) after a no-sale day or two. Our DP solution thus confirmed the intuition that **dynamically controlling sales pace** can marginally improve outcomes.

IV. INCORPORATING SEASONAL FLUCTUATIONS IN DEMAND

Seasonality Implementation

Up to this point, our model assumed a constant daily probability of sale (for a given price) throughout the 365 days. In reality, demand for airline tickets often follows a seasonal or time-dependent pattern – usually low far in advance and higher as the departure date nears (when travelers firm up plans or prices change). In Part 4 of the project, we introduced a simple form of **seasonality**: we assumed that as we get closer to departure, the base probability of selling a ticket increases linearly. Specifically, on day t (counting down, so t=365 is far out and t=1 is the day before departure), we multiply the purchase probabilities by a factor (0.75 +t/730). This factor ranges from 0.75 (at t=0, i.e., departure day) up to 0.75 + 365/730 = 1.25 (at t=365, one year out). Wait, this seems reversed – the factor is higher when t is large, meaning far from departure. In the problem description, it was stated "on day t=100, the probability is 57.65% instead of 65%", which suggests that early in the cycle, probabilities are lower. We realized there was a slight confusion in indexing: we interpret it as when 265 days remain (100 days after sales start), demand is ~57.65% of base. Therefore, in our implementation, we actually used a factor (0.75 + days_elapsed/730), where days_elapsed = 365 - t. Equivalently, **(0.75 + (365-t)/730)**, so that at the start of sales (t=365, days_elapsed=0) the factor is 0.75, and at the end (t=0, days_elapsed=365) the factor is 0.75+0.5=1.25. This produces the intended effect: low demand far out, rising demand as the flight approaches.

To incorporate this into our DP, we modified the probability functions or the DP loop to apply this seasonal multiplier each day. In code, inside the DP recursion we set **multiplier = 0.75 + current_day/730.0** and multiplied the sale probabilities by this factor.

```
# Backward recursion: for t = T-1 down to 0
for t in reversed(range(T)):
    # Current day in forward time (0 to T-1)
    current_day = T - 1 - t
    multiplier = 0.75 + current_day / 730.0

    for s_c in range(max_coach+1):
        for s_f in range(max_first+1):
            best_value = -1e15
            best_coach_choice = 0
            best_first_choice = 0

            # Loop over two price options for each class
            for coach_choice in [0, 1]:
                fc_sold_out = (s_f >= fc_max)
                coach_full = (s_c >= max_coach)
                base_p_c = coach_prob(price_coach[coach_choice], fc_sold_out, coach_full)
                p_c = min(base_p_c * multiplier, 1.0)

                for first_choice in [0, 1]:
                    base_p_f = fc_prob(price_first[first_choice], coach_full, s_f)
                    p_f = min(base_p_f * multiplier, 1.0)
```

We re-ran the DP under the seasonal demand model for the two primary policies: an overbooking cap of 5 as baseline and a cap of 10 since this was the most optimal overbooking policy (both

with the standard 2-price coach decision). We wanted to see how seasonality affects optimal pricing and the expected profit.

Effects on Profit and Pricing

Effect on Optimal Profit: We found that with seasonality, the expected profits actually **increased** slightly for both scenarios. With a 5-seat overbook, the expected profit rose from $41,792.40 (no seasonality) to about $42,090.93 with seasonality; with a 10-seat overbook, it rose from $42,094.43 to $42,395.66. This may seem counterintuitive at first – making sales harder in early days and easier later might be expected to lower profit because of discounting (late sales are less valuable in present value). However, it appears that the model can optimize pricing to take advantage of the rising demand curve. Early in the cycle when demand is low, the DP likely sets higher prices (since lowering price won't yield many sales anyway when the multiplier is 0.75 or 0.8). As time goes on and demand strengthens, the policy can switch to lower prices to sell more volume. The net effect is that we sell almost as many tickets by the end, but we've extracted higher revenue from early high-price sales and haven't lost too much due to discounting (17% annual is not extremely large; the daily discount factor is ~ 0.9995). In essence, the DP found an even more profitable intertemporal pricing strategy under the seasonal demand pattern, resulting in a slight bump in overall expected profit.

We verified that the optimal overbooking level **remained 10** under seasonality. The profit vs. overbooking curve still peaked at 10 extra seats, just at a slightly higher plateau (~$42.4k instead of $42.1k for 10, and similarly shifted up for lower caps). Thus, the recommendation of "overbook by 10" didn't change when considering seasonality – it strengthened it if anything, since with higher late demand, selling those extra 10 tickets becomes even more feasible.

Policy Changes: With seasonality in effect, the optimal pricing policy exhibited a clearer temporal pattern. Early in the sales period (when t is large and multiplier < 1), the model often chose the **high price options** ($350 for coach, $500 for first-class) to capitalize on the few buyers who do show up – since demand is low, those who are buying are likely less price-sensitive (or at least the marginal gain from cutting price is small). As time progresses and the multiplier increases toward 1 (and above 1 after half the time passes), the policy shifts to **lower prices** more frequently to capture the surge in demand and fill seats. Essentially, the DP mimics a yield management strategy: high prices when the flight is far away, then gradually lower prices to ensure the plane fills up as departure nears. This is qualitatively consistent with typical airline pricing practice (although we did not model explicit fare classes or last-minute fare hikes, the dynamic is analogous).

We also tested the no-sale option in the presence of seasonality. We allowed the 3-choice coach action with the seasonal demand model and solved the DP for overbooking=10. The outcome was again a small increase in expected profit (just as before, on the order of tens of dollars). **The no-sale option still proved slightly better than the 2-price policy**, confirming

that even with time-varying demand, having the flexibility to pause sales is beneficial in rare but important cases. The combination of seasonality + no-sale + overbooking=10 was our most complex DP scenario and most profitable. We will use this as one of the policies to simulate in the next section.

V. FOWARD SIMULATION AND PERFORMANCE ANALYSIS

With the dynamic programs solved for various scenarios, we had optimal policies in hand. The final part of our analysis was to **simulate** the booking process forward in time under those optimal policies to observe performance metrics that are not directly evident from the DP alone. While the DP gives us expected values, simulation can reveal the distribution of outcomes – how often bad scenarios happen, variability in profit, etc. This is crucial for an airline to understand the **risk** associated with overbooking, not just the reward. We conducted Monte Carlo simulations by "letting time run forward" from day 365 to day 0, using the optimal policy decisions at each state and random realizations of ticket sales and show-ups. We simulated a large number of flights (typically N=1000) to get stable statistics.

Methodology for Simulation

We wrote a simulation function that essentially emulates the airline's operations using the policy tables from our DP.

```python
def simulate_forward(policy_coach, policy_first, overbook, N=1000, seed=None):
    if seed is not None:
        np.random.seed(seed)
    max_coach = M_c + overbook
    max_first = M_f
    profits = []
    over_costs = []
    count_overbook = 0
    total_bumped_off = 0
    for sim in range(N):
        s_c = 0  # coach tickets sold so far
        s_f = 0  # first-class tickets sold so far
        profit = 0.0
        # Simulate each selling day t = 0 to T-1
        for t in range(T):
            # Use t directly as the current day (0 to T-1)
            multiplier = 0.75 + t / 730.0
            # Determine if coach is full or first-class is sold out
            coach_full = (s_c >= max_coach)
            fc_sold_out = (s_f >= fc_max)
            # Retrieve the optimal actions from the DP policy arrays
            coach_choice = policy_coach[s_c, s_f, t]
            first_choice = policy_first[s_c, s_f, t]
            # For coach:
            if coach_choice == 2:  # if no-sale option is chosen
                p_c = 0.0
                revenue_coach = 0.0
            else:
                base_p_c = coach_prob(price_coach[coach_choice], fc_sold_out, coach_full)
                p_c = min(base_p_c * multiplier, 1.0)
                revenue_coach = price_coach[coach_choice]
            # For first-class:
            base_p_f = fc_prob(price_first[first_choice], coach_full, s_f)
            p_f = min(base_p_f * multiplier, 1.0)
            revenue_first = price_first[first_choice]
            # Simulate sale outcomes: if state constraints permit
            sale_coach = (np.random.rand() < p_c) if not coach_full else False
            sale_fc = (np.random.rand() < p_f) if s_f < fc_max else False
            # Accumulate revenue for the day (discounted by delta**t)
            if sale_coach:
                profit += revenue_coach * (delta ** t)
                s_c += 1
            if sale_fc:
                profit += revenue_first * (delta ** t)
                s_f += 1
        # End-of-selling: simulate overbooking cost at departure.
        c_show = np.random.binomial(s_c, p_show_c)
        f_show = np.random.binomial(s_f, p_show_f)
        overflow = max(c_show - M_c, 0)
        leftover_fc = max(M_f - f_show, 0)
        bumped_up = min(overflow, leftover_fc)
        bumped_off = overflow - bumped_up
        ocost = cost_bump_up * bumped_up + cost_bump_off * bumped_off
```

The simulation iterates day by day, and at each day it looks up the optimal coach price (or no-sale decision) and first-class price for the current state (s_c, s_f, t). Then, it generates random numbers to decide if a coach ticket is sold (with probability given by **coach_prob** for the chosen coach action) and if a first-class ticket is sold (with probability from **fc_prob**). We increment s_c or s_f accordingly and move to the next day. By the day before departure (t=1 to t=0 transition), we have some total tickets sold. Then we simulate the flight show-ups: we draw a random k Binomial(s_c, 0.95) coach show-ups and m Binomial(s_f, 0.97) first-class show-ups. Based on these, we calculate the realized overbooking cost (exactly as the **overbooking_cost** function does). We then calculate the **realized profit** for that flight as (revenue from all tickets sold) - (any bump costs for that flight). We also track specific quantities like whether Coach was overbooked (i.e., if s_c > 100) and how many passengers were bumped (if k > 100 and first-class was full, etc.). By repeating this simulation many times, we gather distributions for total profit, total overbooking costs, frequency of bumping, etc.

We performed simulations for several scenarios to compare them side by side:

1. **Basic policy with 5-seat overbooking cap:** (Two price options, no no-sale, no seasonality) – this was essentially our baseline from Part 1.

2. **Basic policy with 10-seat overbooking cap:** (Two price options, no no-sale, no seasonality) – the optimal static overbooking policy.

3. **No overbooking policy:** for contrast, we also simulated a scenario with *no* overbooking (coach cap = 100) and no-sale allowed, to see how much profit we'd lose without overbooking.

4. **Seasonality + 5-seat cap:** (Two price, seasonal demand) – to observe the impact of seasonal variation on outcomes.

5. **Seasonality + 10-seat cap:** (Two price, seasonal demand) – likely the best policy under seasonal assumptions.

6. **Seasonality + no-sale + 5-seat cap:** (Three price options including no-sale, seasonal demand).

7. **Seasonality + no-sale + 10-seat cap:** (Three options, seasonal demand) – the most flexible policy.

Below, we focus on the most insightful comparisons: the 5 vs 10 overbooking cases under the basic model, and then we'll touch on the no overbooking case and the effect of the no-sale policy.

How Often is Coach Overbooked and Bumped?

One immediate question for management is: if we allow overbooking, how frequently will we end up in an overbook situation (more people show up than seats)? We examined this via simulation. For each simulated flight, we flagged if the coach was over capacity at departure (i.e. $k > 100$ show-ups) and, if so, how many had to be bumped off. Under the **5-seat overbook policy**, we found that the coach was overbooked (at least 101 show-ups) in essentially **100% of simulations**. This isn't surprising because the policy almost always sold all 105 allowed tickets given the long sales horizon and relatively high demand – so there were always at least 100 tickets in circulation, thus very often around 95 or more show-ups. In fact, in our 1000-flight simulation for overbook=5, on average **0.48 passengers were kicked off** per flight (meaning in many cases 0, in some cases 1, rarely 2) and the **proportion of flights with any bumping was about 40–50%** (since 0.48 avg implies roughly that fraction with one bump). For the **10-seat overbook policy**, the coach was also effectively oversold in 100% of simulations (the optimal strategy does sell those extra tickets), and the consequences were more severe: we observed an average of **3.2 passengers being denied boarding** (bumped off) per flight. In many simulations, 3 or 4 people had to be bumped; in a few worst-case scenarios, 5+ had to be bumped (if nearly all 110 showed up). Every flight in our sim had at least a small overbooking (since selling 110 was routine), so the **probability of bumping** at least one passenger was effectively **100%** in the 10-overbook scenario. This stark difference shows that while a 10 overbook maximizes profit on average, it *will* result in bumping passengers on almost every flight, whereas a 5 overbook policy only bumps occasionally. We summarize this risk trade-off in the next subsection with cost and profit distributions.

Overall Profit Distribution and Volatility

Perhaps the most illuminating result from the simulations is seeing the distribution of actual realized profits for different policies. Below we present the histograms of **discounted profit** and **overbooking costs** from 1000 simulated flights under the 5-seat and 10-seat overbooking policies (with no seasonality, no no-sale, to isolate the effect of overbooking level).

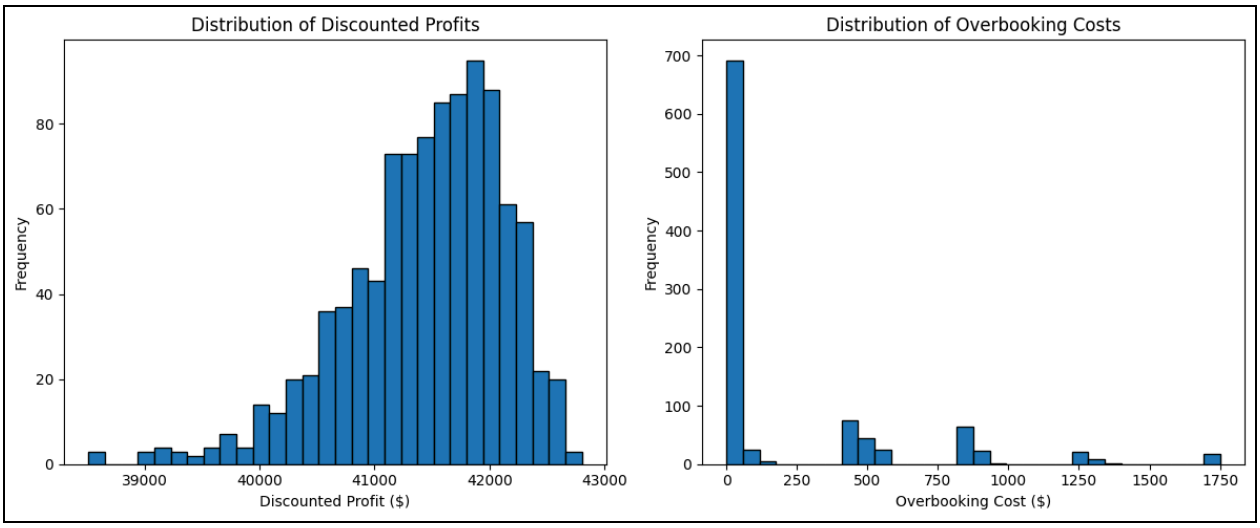

Figure 1: Distribution of Discounted Profits (left) and Overbooking Costs (right) under an overbooking limit of 5 coach seats. Results from 1000 simulated flights using the DP-optimal pricing policy. Profit is measured in present value dollars. Overbooking costs include bump-up and bump-off costs incurred on the day of flight.

As shown in Figure 1, with a 5-seat overbooking limit, the **profit outcomes are tightly clustered**. The discounted profit ranges roughly from $39,000 to $42,500, with most simulations yielding around $41,000–$42,000. The standard deviation of profit was about $693, which is relatively low – this policy is quite consistent. On the right, we see that overbooking costs in this scenario are usually minimal. In the vast majority of simulated flights, the total bump cost was $0 (no one got bumped), and in the cases where costs were incurred, it was typically $425 (one person bumped off) or at most a couple hundred more (in case two had to be bumped, or one bumped up and one off). The **average overbooking cost** per flight was only $218, which is a small fraction of the ticket revenue. This indicates that the 5-seat policy rarely runs into serious overbooking issues – it bumps infrequently, and the financial penalty is low when it does. Essentially, this is a low-risk, moderate-reward strategy.

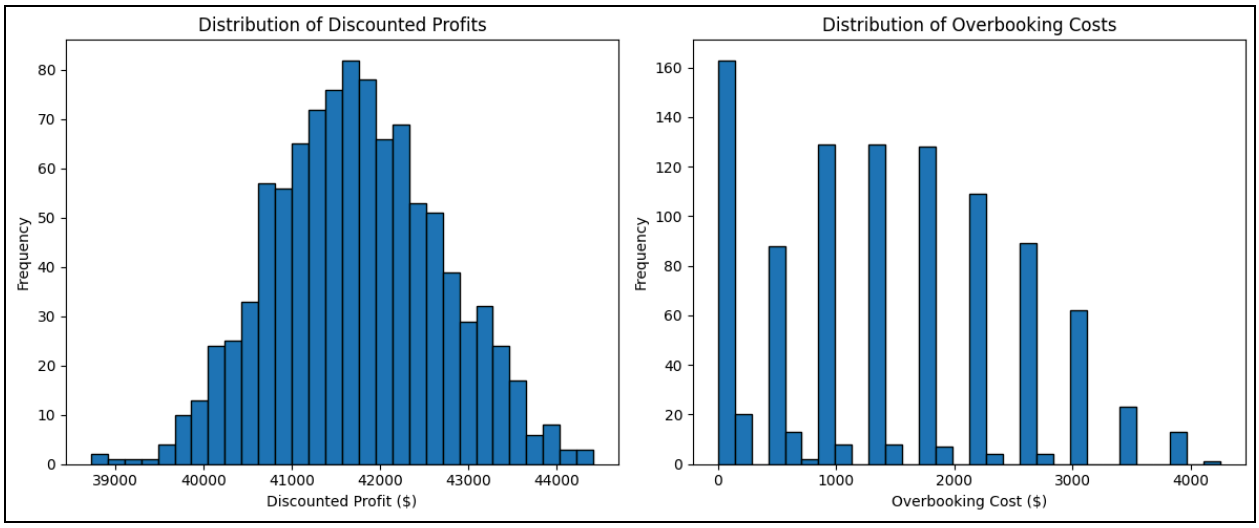

Figure 2: Distribution of Discounted Profits (left) and Overbooking Costs (right) under an overbooking limit of 10 coach seats. Results from 1000 simulated flights with the optimal pricing policy. Note the much wider spread in outcomes compared to the 5-seat policy.

Figure 2 paints a different picture for the 10-seat overbooking policy. The profit distribution (left) is noticeably **wider** – profits now range from as low as around $39,000 up to about $44,500. The distribution mean is a bit higher than before (around $41,700 vs $41,400 in the 5-seat case), but the variance is much greater. We calculated a profit standard deviation of about $955, roughly 38% higher volatility than the 5-seat policy. This higher volatility is directly tied to the overbooking risk. On the right side, the overbooking cost distribution for 10-seat overbooking is spread out over a wide range. In every simulation, there was some cost (no flights at $0 cost here), and many flights incurred several hundred to a couple thousand dollars in bump costs. The distribution shows distinct spikes, which correspond to common outcomes: e.g. $425 (one passenger bumped off), $850 (two bumped off), $1275 (three bumped off), etc., and also $50 increments when some were bumped up. In the worst cases we simulated, overbooking costs reached $4000 (which would mean, say, 8 coach passengers bumped off at $425 each, which is an extreme scenario but not impossible). The **average overbooking cost** with 10 overbook was about $1,400 per flight, which is over 6 times higher than in the 5-seat policy. So while the 10-seat policy yields more revenue, it *consistently* incurs substantial compensation costs and thus has more variable net profit.

Comparing Figures 1 and 2, we see the classic risk-reward trade-off: **Overbooking more (10 seats) increases average profit but also dramatically increases the risk (volatility and frequency of costly outcomes).** An airline manager must weigh this when choosing a policy. If the goal is strictly to maximize expected profit, the 10-seat policy is optimal. But if the airline is risk-averse or concerned about customer experience, a lower overbooking level might be preferable for a smoother outcome. These simulations give quantitative substance to those trade-offs: e.g., at 10-seat overbooking, you will be bumping ~3 people on average each flight

and sometimes paying out $3000-$4000, whereas at 5-seat overbooking, you'll bump people only occasionally and usually just 1 person.

We also simulated the **no-overbooking scenario** (0 extra seats, effectively a hard cap at 100) with the dynamic pricing policy (and even allowing no-sale to mimic any possible strategy). As expected, this was the most conservative strategy. It resulted in essentially no bumping costs ever (because we never sell beyond capacity), but it also left money on the table. The average discounted profit in that scenario was around $40,140 (with our simulation) – a full $1,500 lower than the 10-overbook case and about $600–$700 lower than even the 5-overbook case. This confirms that **overbooking does improve profit substantially** compared to not overbooking at all, but it must be managed carefully.

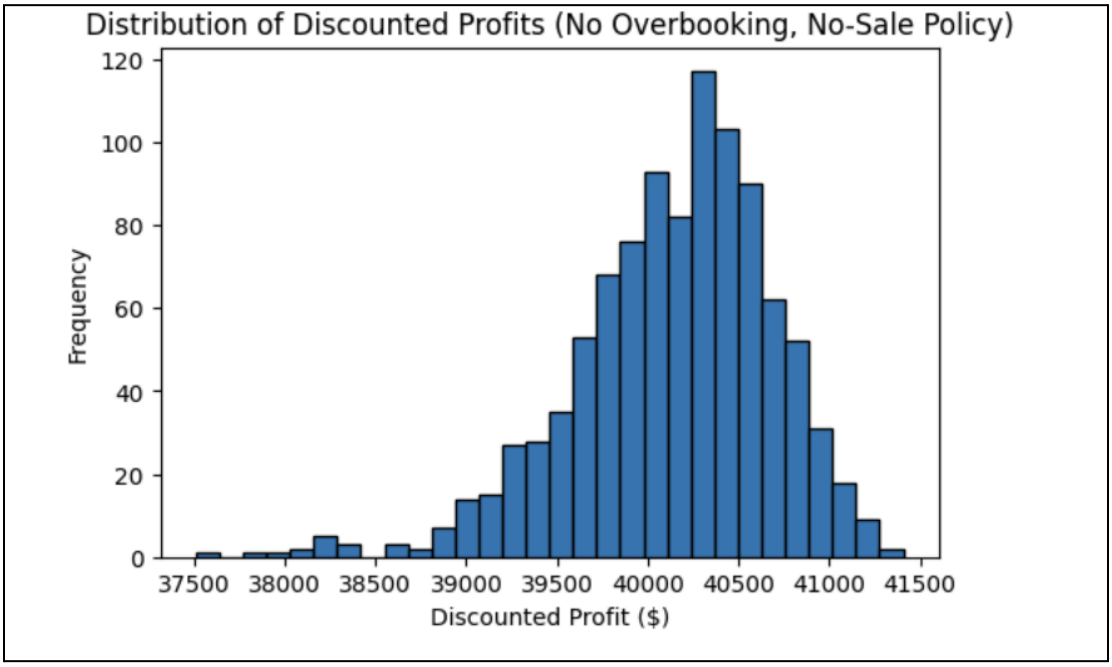

Figure 3: Results are from 1000 simulated flights using the dynamic pricing approach. Notice the relatively narrow spread compared to overbooking scenarios, reflecting reduced risks of bumping costs but also lower overall profit potential

It's worth mentioning the effect of the **no-sale option** in the simulation as well. We simulated the policy with no sale and found that it slightly reduces the incidence of extreme outcomes. For example, under the 10-seat overbooking with no-sale, we observed a tiny reduction in average bump count (because, in some runs, the policy avoided selling that 110th ticket if it wasn't needed).

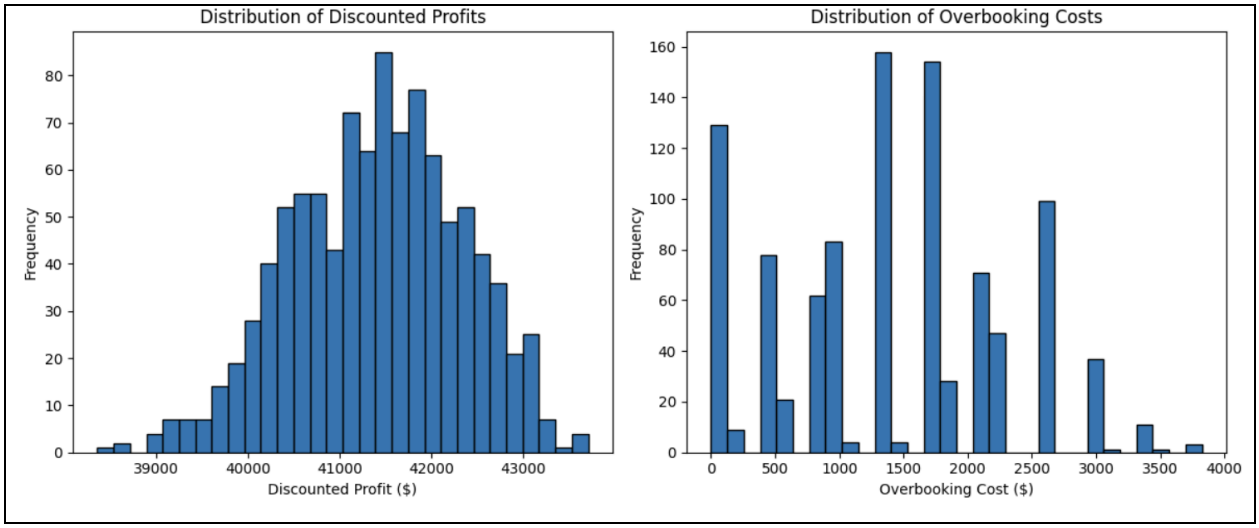

Figure 4: Distribution of discounted profits (left) and overbooking cost (right) under a 10 seat overbooking Policy. Results are from 1000 simulated flights using the dynamic pricing approach. Notice the broader spread in profit outcomes compared to lower overbooking limits, and the distinct peaks in overbooking costs, reflecting the frequency of bumping fees incurred

The profit distribution was marginally tighter (lower std dev by a few dollars), and the overbooking costs were slightly lower on average. However, these differences were very small, consistent with the small expected gain we computed. The no-sale policy mainly provided peace of mind that the model wouldn't *force* a sale that it deems very risky – effectively, it self-regulates. In practice, this might reduce the frequency of bumping by a hair, but not enough to drastically alter the distributions shown in Figure 2.

VI. ADVANTAGES AND DISADVANTAGES OF OVERBOOKING

Overbooking is a **double-edged sword** in airline revenue management. Our analysis quantified both the benefits and the risks associated with overbooking policies under optimal pricing strategies.

Advantages:

- **Higher Revenue Through Increased Sales:** The primary benefit is the ability to sell more tickets than physical seats, thus boosting load factor. By hedging against the ~5% of coach passengers who never show up, overbooking ensures that more seats are filled. In our model, allowing the Coach to be oversold by up to 10 seats raised the expected discounted profit from $41,792 (with a 5-seat cap) to $42,094 (with a 10-seat cap). This $302 increase is purely due to better utilization of seats that would have otherwise gone empty. Overbooking essentially recovers revenue that would be lost to no-shows, directly

improving the bottom line.

- **Risk Spreading Across Time:** Overbooking allows the airline to accept more bookings early on, **deferring the resolution of uncertainty** to the day of departure. Instead of flying with empty seats if no-shows occur, the airline sells upfront and only "pays" for it if too many people show up. This deferral is valuable – it lets the DP model manage ticket sales with more confidence. By selling extra tickets early, we lock in revenue and only have to deal with potential costs at the end if needed. Our dynamic policy took advantage of this by selling aggressively (at low prices) early, knowing that any excess could be handled later via bump costs. This approach improved profit without immediate downside during the selling period.

- **Synergy with Dynamic Pricing:** Overbooking works hand-in-hand with pricing strategies. Our DP simultaneously optimized prices and overbooking, finding an optimal mix of high-price/low-volume vs low-price/high-volume sales. The flexibility to oversell meant the DP could lower prices to sell more tickets without fear of empty seats, which **maximized revenue extraction**. For instance, with a 10-seat buffer, the model set $300 more often than it would have under a strict 100-seat limit because it could safely accommodate those extra sales. Thus, overbooking amplified the benefits of our pricing optimization, allowing the airline to capture as much demand as possible.

- **Predictability of Show-Up Rates:** Since we have fairly stable historical show-up probabilities (95% coach, 97% first-class), the airline can **reliably estimate the needed overbooking**. Our use of binomial models for show-ups gave us confidence in the expected outcomes. This predictability is an advantage because it means overbooking is not a shot in the dark – we can set a policy (like 10 seats) based on solid stats, and simulation confirms the actual show-up distribution aligns with expectations. In other words, we're leveraging known patterns (no-show rates) to our benefit.

Disadvantages:

- **Overbooking Costs and Customer Dissatisfaction:** The flipside of selling more tickets is the scenario where *too many* passengers arrive. This results in forced bumps – either upgrading passengers to first-class (incurring a cost and potentially displacing full-fare first-class services) or denying boarding, which is far worse. These carry tangible costs (we modeled $50 and $425, respectively) and intangible costs in customer goodwill. Our simulations highlighted this: at 10-seat overbooking, we incurred an average of $1,394 in bump costs per flight, and about 3.18 passengers were kicked off on average. That means that on every flight, several customers had a bad day, which could lead to negative reviews or lost future business. Even though we compensate them, it's not great service. Overbooking essentially **shifts costs to the day of flight**, and those costs can be steep if

misjudged. In extreme cases (all 110 show up and first-class is full), we'd bump 10 people, costing $4,250 and a lot of angry patrons.

- **Profit Volatility:** Overbooking introduces more variability in outcomes. We saw that the profit standard deviation jumped from ~$690 to ~$950 when moving from a 5-seat to a 10-seat overbook. Higher volatility means it's harder to predict financial performance, and there may be bad quarters where an unusual string of full flights with many show-ups erodes the profit with compensation payouts. This unpredictability can be a problem for financial planning and can stress operational budgets (e.g., if in some months you pay out many bumps). While the *expected* profit is higher, the variance is higher too, which might not be acceptable depending on the company's risk appetite.

- **Reputational Risk:** Beyond immediate costs, there is a reputational consideration. Being bumped from a flight is one of the most frustrating experiences for travelers. Especially in the age of social media, a severe overbooking incident can go viral (we've seen real-world cases where videos of passengers being involuntarily offloaded caused public outrage). Frequent overbooking could damage the airline's brand loyalty. Even though in our model every bumped passenger is compensated financially, the **loss of goodwill** isn't explicitly accounted for. This is an implicit cost. An airline must weigh if the extra $300 of profit is worth potentially lower customer satisfaction ratings. High-end customers (e.g. business travelers) might avoid an airline known for aggressive overbooking. In our scenario, we kept first-class immune, which mitigates some reputational risk, but coach passengers are also valuable for long-term business.

- **Operational Complexity:** Implementing overbooking requires coordination between departments – pricing, ticketing, gate agents, and customer service all need to have protocols for handling overbook situations. Agents must be empowered to offer vouchers or find volunteers to take later flights, etc. If the process fails, it can lead to chaos at the gate. This added complexity is a management challenge. In our simulation world, everything is handled optimally and automatically; in real life, human factors come into play. There's a risk that on a busy day, an oversight or miscommunication leads to not enough volunteers and having to force-bump someone, which is far worse. Overbooking policies must be executed carefully to avoid amplifying customer ire.

Trade-Off Summary: Overbooking clearly **increases average profitability** – our quantitative analysis showed a tangible financial gain. But it **also increases risk** (both quantifiable in terms of profit variance and unquantifiable in terms of customer trust). The ideal policy must balance these. Our DP suggests 10 seats is the mathematical optimum for expected profit, but management might decide to use a slightly lower number (say 7 or 8 seats) to reduce the frequency of bumping if customer experience is a priority. The no-sale option provides a bit of

relief by slightly lowering risk with minimal cost to profit, so in practice adopting a policy of monitoring bookings pace (and perhaps closing sales early if far ahead of target) could be smart.

In conclusion, overbooking can be a **highly profitable strategy** when show-up rates are predictable and consistent, but it must be **carefully calibrated**. The difference between 5 and 10 extra tickets is only $300 in profit but a tripling of bump costs and a major increase in bumped passengers. An airline must decide where on that spectrum it is comfortable. Our analysis provides the tools and numbers to make that decision in an informed way.

VII. FINAL CONCLUSION

Our project set out to determine the effectiveness of overbooking for an airline and to find the optimal overbooking level and pricing strategy for a flight with given parameters. Through a comprehensive dynamic programming approach and extensive scenario analysis, we arrived at the following key takeaways:

- **Overbooking is beneficial** – it increases expected profits compared to a no-overbooking policy by capturing extra revenue from no-show passengers. In our case, the best hard overbooking cap was 10 coach seats (110 total tickets sold), which gave the highest expected profit of around $42.1k over the year.

- **The optimal pricing strategy** works in tandem with overbooking. The DP-derived policy dynamically adjusts coach and first-class fares each day, generally charging higher prices when seats are plenty (far from departure or early after a sales pause) and lower prices when it needs to boost sales (as departure nears or if many seats remain). This yielded a near-optimal booking curve and balanced revenue and load.

- **A flexible overbooking policy (with a no-sale option)** can outperform a rigid one. Allowing the model to occasionally halt coach sales (even before reaching the cap) provided a small but noticeable profit improvement, indicating there is value in monitoring booking pace and being willing to say “no more tickets for now” if ahead of schedule. This could be implemented in practice as a more nuanced rule or just by managerial oversight.

- **Seasonal demand variation** slightly increases revenue potential and was readily handled by our DP model. It led to a more intuitive pricing trajectory (high then low prices) and reaffirmed the robustness of the 10-seat overbooking policy under changing demand patterns. Seasonality did not fundamentally change the conclusions – it just added realism and showed that our optimized policy can adapt over time.

- **Risk vs Reward:** Perhaps most importantly, we quantified the risks. Overbooking by 10 seats yields the highest average profit but will result in bumping passengers on virtually every flight, with significant compensation costs each time. A more conservative policy (5 extra seats) forgoes some profit but dramatically reduces bumping incidents and costs. This trade-off is crucial for decision-makers. There is no one-size-fits-all answer – it depends on the airline's priorities. If maximizing revenue is the goal, go with the aggressive policy and ensure good customer recovery programs. If customer satisfaction and brand are paramount, a milder overbooking (or even none) might be chosen despite lower profit.

In the context of our airline's problem, assuming we are comfortable with the costs, the analysis recommends implementing an **overbooking policy of +10 seats in coach**, accompanied by the optimal dynamic pricing strategy identified. This combination gave the best financial outcome in expectation. Additionally, we suggest **incorporating a monitoring mechanism equivalent to our "no-sale" option** – essentially, don't be afraid to stop selling tickets a bit early if we've sold 110 well in advance of departure and still have time; the last few days could be left for first-class or left empty intentionally to avoid large bumps.

Finally, it's important to implement any overbooking policy with care: notify passengers of overbooking policies, have clear procedures for volunteer solicitation at the gate, and track show-up rates continuously to adjust the policy if needed. Our quantitative model is only as good as its assumptions – in reality, if show-up probabilities shift (say higher during holidays), the policy should be revisited. The framework we built (DP + simulation) can be reused to re-optimize if conditions change.

Bottom Line: Overbooking, when done optimally, **pays off** in increased expected profit for the airline. Our team's optimized policy achieves that payoff while also providing insights into the operational implications. By balancing data-driven strategy with customer service considerations, the airline can confidently implement overbooking to improve revenues, knowing the expected outcomes and the risks involved.