

Airline Case Study - Final Report

Yaseen Tolba, Caton Zhu, Elias Badway

IE 370 - Spring 2025

I. Problem Statement and MDP Formulation	2
Flight Details and Parameters	2
1.1 MDP Components	2
State Space	2
Action Space	2
Expected Revenue Table	3
1.2 State Transition Probabilities	3
Appendix A: Expected Revenue Calculations	3
Action (0,0)	3
Action (1,0)	3
Action (0,1)	3
Action (2,0)	4
Action (1,1)	4
Action (0,2)	4
Appendix B: Transition Probability Calculations	4
Action (0,0)	4
Action (1,0)	4
Action (0,1)	4
Action (2,0)	5
Action (1,1)	5
Action (0,2)	5
Conclusion	5
Appendix C: Python Code for Optimal Policy Calculation	5
Optimal Earnings Table	7
Optimal Actions Table	7

I. Problem Statement and MDP Formulation

We analyze an airline seat-inventory control problem over a finite horizon using a Markov Decision Process (MDP) framework. The objective is to maximize expected revenue from selling flight tickets over a limited sales window, while considering stochastic demand and customer behavior.

Flight Details and Parameters

- Total Seat Capacity: 15 seats
- Sales Horizon: 11 days, from May 5 (Day 0) to May 15 (Day 10)
- Daily Ticket Demand X : $P(X = 0) = 1/4$, $P(X = 1) = 1/2$, $P(X = 2) = 1/4$
- Ticket Types:
 - Cheap ticket: \$100
 - Expensive ticket: \$200
- Each day, up to 2 tickets can be offered: action (p, q) , with p = cheap tickets, q = expensive tickets. Constraint: $p + q \leq \min(2, i)$, where i is seats remaining.
- Conversion: Excess demand may convert to expensive tickets with probability 0.4
- Goal: Maximize expected total revenue over 11 days
- Discount Factor: $\alpha = 1$

1.1 MDP Components

State Space

State $s_t = i \in \{0, 1, \dots, 15\}$ = seats remaining on day $n \in \{0, \dots, 10\}$.

Action Space

Each day, select action $a_t = (p, q) \in A$:

$A = \{(0,0), (1,0), (0,1), (2,0), (1,1), (0,2)\}$, subject to $p + q \leq \min(2, i)$

Expected Revenue Table

Expected revenue $g(p,q)$ for each action, based on daily demand distribution:

Action (p,q)	Expected Revenue $g(p,q)$
(0,0)	\$0
(1,0)	\$75
(0,1)	\$60
(2,0)	\$100
(1,1)	\$95
(0,2)	\$80

1.2 State Transition Probabilities

Transition probabilities from state i to i , $i-1$, or $i-2$ for each action:

Action (p,q)	$P(Y = 0)$	$P(Y = 1)$	$P(Y = 2)$
(0,0)	1.00	0.00	0.00
(1,0)	0.25	0.75	0.00
(0,1)	0.70	0.30	0.00
(2,0)	0.25	0.50	0.25
(1,1)	0.25	0.65	0.10
(0,2)	0.64	0.32	0.04

Appendix A: Expected Revenue Calculations

Action (0,0)

Revenue = 0 for all X

$$E[g(0,0)] = 0.25 \times 0 + 0.5 \times 0 + 0.25 \times 0 = \$0$$

Action (1,0)

If $X=0$: 0 sold $\rightarrow \$0$

If $X=1$ or 2 : 1 cheap $\rightarrow \$100$

$$E[g(1,0)] = 0.25 \times 0 + 0.5 \times 100 + 0.25 \times 100 = \$75$$

Action (0,1)

If $X=0$: 0 sold $\rightarrow \$0$

If $X=1$: 1 tries expensive (0.4) $\rightarrow \$80$

If $X=2$: 2 try $\rightarrow E = 0.8 \times 200 = \160

$$E[g(0,1)] = 0.25 \times 0 + 0.5 \times 80 + 0.25 \times 160 = \$60$$

Action (2,0)

If $X=0$: 0 sold $\rightarrow$ \$0

If $X=1$: 1 sold $\rightarrow$ \$100

If $X=2$: 2 sold $\rightarrow$ \$200

$$E[g(2,0)] = 0.25 \times 0 + 0.5 \times 100 + 0.25 \times 200 = \$100$$

Action (1,1)

If $X=0$: 0 sold $\rightarrow$ \$0

If $X=1$: 1 cheap $\rightarrow$ \$100

If $X=2$: 1 cheap + 1 tries expensive (0.4) $\rightarrow 100 + 80 = \$180$

$$E[g(1,1)] = 0.25 \times 0 + 0.5 \times 100 + 0.25 \times 180 = \$95$$

Action (0,2)

If $X=0$: 0 sold $\rightarrow$ \$0

If $X=1$: 1 tries expensive (0.4) $\rightarrow$ \$80

If $X=2$: $P(2) = 0.16$, $P(1) = 0.48$, $P(0) = 0.36 \rightarrow E = \160

$$E[g(0,2)] = 0.25 \times 0 + 0.5 \times 80 + 0.25 \times 160 = \$80$$

Appendix B: Transition Probability Calculations

Action (0,0)

Always 0 sold $\rightarrow P(i \rightarrow i) = 1.0$

Action (1,0)

If $X=0$: $Y=0 \rightarrow 0.25$

If $X \geq 1$: $Y=1 \rightarrow 0.75$

$P(i \rightarrow i) = 0.25$, $P(i \rightarrow i-1) = 0.75$

Action (0,1)

If $X=0$: $Y=0 \rightarrow 0.25$

If $X=1$: 1 tries expensive (0.4) $\rightarrow 0.5 \times 0.4 = 0.2$

If $X=2$: $P(2)=0.16$, $P(1)=0.48$, $P(0)=0.36$

Final: $P(i \rightarrow i) = 0.64$, $P(i \rightarrow i-1) = 0.32$, $P(i \rightarrow i-2) = 0.04$

Action (2,0)

$X=0: Y=0 \rightarrow 0.25$

$X=1: Y=1 \rightarrow 0.5$

$X=2: Y=2 \rightarrow 0.25$

$P(i \rightarrow i)=0.25, P(i \rightarrow i-1)=0.5, P(i \rightarrow i-2)=0.25$

Action (1,1)

$X=0: Y=0 \rightarrow 0.25$

$X=1: Y=1 \rightarrow 0.5$

$X=2: Y=2$ with $0.25 \times 0.4 = 0.1$, $Y=1$ with $0.25 \times 0.6 = 0.15$

$P(i \rightarrow i)=0.25, P(i \rightarrow i-1)=0.65, P(i \rightarrow i-2)=0.10$

Action (0,2)

$X=0: Y=0 \rightarrow 0.25$

$X=1: 1 \text{ tries} \rightarrow 0.4 \text{ success} \rightarrow 0.5 \times 0.4 = 0.2$

$X=2: P(2)=0.16, P(1)=0.48, P(0)=0.36$

Final: $P(i \rightarrow i)=0.64, P(i \rightarrow i-1)=0.32, P(i \rightarrow i-2)=0.04$

Conclusion

In this report, we formulated the airline seat-inventory problem as a finite-horizon Markov Decision Process (MDP), capturing the trade-offs between immediate revenue from cheap and expensive tickets and the stochastic nature of customer demand. By enumerating the expected revenues and transition probabilities for all feasible ticket allocation actions, we created a comprehensive framework for optimizing daily decisions.

Using this optimized strategy, the airline can expect to achieve a maximum total revenue of \$1097.24 over the 11-day sales window.

Appendix C: Python Code for Optimal Policy Calculation

```
import numpy as np

def calculate_optimal_policy():
    num_states = 16
    days = 11
    action_codes = ["(0,0)", "(1,0)", "(0,1)", "(2,0)", "(1,1)", "(0,2)"]
```

```

revenue = [0, 75, 60, 100, 95, 80]

transitions = np.zeros((6, num_states, num_states))
for s in range(num_states): transitions[0][s][s] = 1.0
for s in range(1, num_states): transitions[1][s][s] = 0.25; transitions[1][s][s-1] = 0.75
for s in range(1, num_states): transitions[2][s][s] = 0.7; transitions[2][s][s-1] = 0.3
for s in range(2, num_states):
    transitions[3][s][s] = 0.25; transitions[3][s][s-1] = 0.5; transitions[3][s][s-2] = 0.25
    transitions[4][s][s] = 0.25; transitions[4][s][s-1] = 0.65; transitions[4][s][s-2] = 0.1
    transitions[5][s][s] = 0.64; transitions[5][s][s-1] = 0.32; transitions[5][s][s-2] = 0.04

values = np.zeros((days, num_states))
strategy = np.zeros((days, num_states), dtype=int)
for day in reversed(range(days)):
    for seats in range(num_states):
        best_value = 0; chosen_action = 0
        for action in range(6):
            valid = (action == 0) or (action in [1,2] and seats >= 1) or (action in [3,4,5] and seats >= 2)
            if valid:
                current_value = revenue[action]
                if day < days-1:
                    for next_seats in range(num_states):
                        current_value += transitions[action][seats][next_seats] * values[day+1][next_seats]
                if current_value > best_value:
                    best_value = current_value; chosen_action = action
            values[day][seats] = best_value; strategy[day][seats] = chosen_action

return values, strategy, action_codes

def show_results(values, strategy, action_names):
    print("Optimal Earnings:\n" + "-"*85)
    print("Day | " + " ".join(f"{s:5}" for s in range(16)))
    print("-"*85)
    for day in range(11):
        print(f"{day:2} | " + " ".join(f"{v:5}" for v in values[day]))

    print("\nBest Strategies:\n" + "-"*85)
    print("Day | " + " ".join(f"{s:5}" for s in range(16)))
    print("-"*85)
    for day in range(11):
        print(f"{day:2} | " + " ".join(f"{action_names[strategy[day][s]]:5}" for s in range(16)))

if __name__ == "__main__":
    print("Calculating Best Ticket Sales Strategy...\n")
    earnings, plan, actions = calculate_optimal_policy()
    show_results(earnings, plan, actions)
    print(f"\nMaximum Expected Revenue: ${earnings[0][15]:.2f}")

```

Optimal Earnings Table

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0. 0	196. 5	385. 4	554. 9	693. 6	796. 3	867. 3	916. 8	954. 9	988. 7	1020. 2	1047. 6	1069. 1	1083. 9	1092. 8	1097. 2
1	0. 0	195. 0	379. 7	540. 0	664. 9	752. 9	812. 1	854. 4	889. 1	921. 3	949.2	970.9	985.5	993.9	997.8	999.4
2	0. 0	192. 8	371. 9	520. 8	630. 4	703. 7	752. 7	789. 7	822. 3	850. 8	872.7	887.1	894.9	898.4	899.6	899.9
3	0. 0	189. 7	361. 3	496. 2	589. 3	648. 9	689. 8	723. 4	752. 6	774. 8	788.8	796.0	798.9	799.8	800.0	800.0
4	0. 0	185. 3	346. 8	465. 3	541. 4	589. 2	624. 4	654. 5	677. 0	690. 6	697.1	699.3	699.9	700.0	700.0	700.0
5	0. 0	179. 0	327. 4	427. 1	486. 9	525. 4	556. 6	579. 4	592. 5	598. 1	599.7	600.0	600.0	600.0	600.0	600.0
6	0. 0	170. 0	301. 6	380. 9	426. 5	458. 9	482. 1	494. 5	498. 9	499. 9	500.0	500.0	500.0	500.0	500.0	500.0
7	0. 0	157. 1	267. 8	326. 4	361. 5	385. 2	396. 5	399. 6	400. 0	400. 0	400.0	400.0	400.0	400.0	400.0	400.0
8	0. 0	138. 8	224. 0	264. 4	289. 1	298. 4	300. 0	300. 0	300. 0	300. 0	300.0	300.0	300.0	300.0	300.0	300.0
9	0. 0	112. 5	168. 8	193. 8	200. 0	200. 0	200. 0	200. 0	200. 0	200. 0	200.0	200.0	200.0	200.0	200.0	200.0
10	0. 0	75.0	100. 0	100. 0	100. 0	100. 0	100. 0	100. 0	100. 0	100. 0	100.0	100.0	100.0	100.0	100.0	100.0

Optimal Actions Table

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	(0,0)	(0,1)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
1	(0,0)	(0,1)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
2	(0,0)	(0,1)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
3	(0,0)	(0,1)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
4	(0,0)	(0,1)	(0,2)	(0,2)	(0,2)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
5	(0,0)	(0,1)	(0,2)	(0,2)	(0,2)	(0,2)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
6	(0,0)	(0,1)	(0,2)	(0,2)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
7	(0,0)	(0,1)	(0,2)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
8	(0,0)	(0,1)	(0,2)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
9	(0,0)	(0,1)	(1,1)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)
10	(0,0)	(1,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)	(2,0)