Destination Unknown: Using MDPs to Optimize Airport Gate Assignment

Vale Rasmussen
Stanford Intelligent Systems Lab
Department of Aeronautics and Astronautics
Stanford, CA
valer22@stanford.edu

Rana Taki
Stanford Intelligent Systems Lab
Department of Mechanical Engineering
Stanford, CA
ranataki@stanford.edu

***Abstract*—This paper demonstrates the benefits of using Markov Decision Processes (MDPs) to model and solve the airport gate assignment problem (AGAP). Each commerical aircraft that arrives at its destination airport worldwide needs to be given a gate assignment, and a variety of stakeholders want to ensure that gates are allocated in the "best" way possible. We first introduce which parameters need to be considered when solving the gate assignment problem such as aircraft delay and airline gate preferences. We then discuss how we created a realistic simulated airport environment to gather data for our MDP solution models. Our stochastic model introduces a realistic amount of uncertainty. Additionally, we implemented an interactive simulation mode, allowing a human operator to assign gates manually for comparison. After gathering simulated data, we found that online MDP solution methods like Q-learning and Monte Carlo Tree Search (MCTS) generated policies that greatly outperformed a random agent. Our findings demonstrate the potential of using MDP methods to manage airports around the world more efficiently.**

I. Introduction and Literature Review

A. Context

At airports around the world, a complex "game" is constantly being played. Ground controllers and dispatchers must assign a gate to each arriving aircraft. Unlike a typical sorting process, choosing this gate can have far-reaching consequences for the operation of the airport. For example, some gates might be preferred by certain airlines, while other gates are preferred by passengers or ground crew. One of the most critical considerations is limiting delays. All stakeholders hope that delays can be as small as possible. Even one delayed flight has the potential to cause a ripple effect where future flights are delayed as well. One study by Rome et al. (2001) estimates that arrival delays caused by this ripple effect is costing airlines approximately $75 million in lost revenue per year.

In real airport operations, there is a significant amount of uncertainty. For example, some aircraft might taxi faster than others, some gates may stay occupied for longer than expected, and much more. Given this uncertainty and the large number of complex stakeholder values discussed above, finding an optimal or even "good" strategy by hand is difficult. Yet, this problem remains extremely enticing because of its ubiquity and high stakes. In academic literature, it has been formalized as *the Airport Gate Assignment Problem* (AGAP).

B. Research Gap

Many authors have proposed approaches to create better polices for airport gate assignment. A survey by Bouras et al. (2014) investigates the state of the art for modeling and attempting to solve the problem. They write that "many researchers formulated the AGAP as an integer, binary, or mixed integer linear or nonlinear model". These models are created by directly modeling rewards and costs as a function of parameters like walking distance, gate size, etc. While this approach can be a reasonable interpretation of the values behind the problem, it effectively only calculates the ideal final state. The dynamics and 'ripple effect' is much harder to capture.

To better incorporate this time complexity, we propose using a Markov Decision Process (MDP) to model the AGAP. The MDP is an ideal choice for modeling and solving sequential decision problems. Once the problem is modeled as an MDP, there are a plethora of robust methods to find approximately optimal policies even in environments with many possible states (Kochenderfer et al. 2022). A previous study by Aoun and Afia (2014) also proposed using MDPs for gate assignment. This research mostly focuses on managing delays that have already occurred in flight.

Our paper expands on this study by introducing realistic simulation of aircraft taxi delays and gate occupancy. While Aoun and Afia (2014) investigated the performance of policy iteration as a solution method for AGAP, our paper will demonstrate the performance of more robust methods such as Q-learning and Monte Carlo Tree Search (MCTS).

II. Tangible Problem Scoping

Before creating our simulation and MDP components, we first determined the scope of our airport environment. As discussed earlier, there are nearly infinite stakeholder values that could be implemented into the gate assignment problem. However, with the limited time to complete our project, we limited our scope to the aspects that we felt were most critical and most interesting. This included stochastic aircraft taxi speeds, certain gates being preferred by certain airlines, and penalties for blocked or delayed movement. Taxiway routing is a complex task in its own regard, so we decided to model a simple airport with only one runway, one taxiway, and one

terminal as shown in Fig. 1. This formulation allowed us to use discrete states to represent aircraft positions, eliminating some complexity. The following section covers how the simulation works in greater detail.

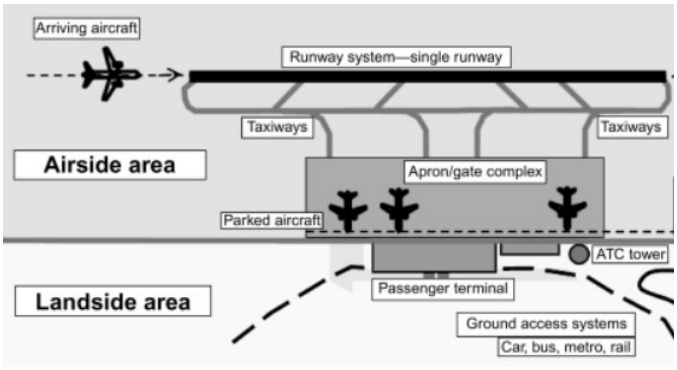

Fig. 1. Visualization of Simple Airport for Modeling

III. Simulation

A. Operation

To create realistic MDP states, actions, transitions, and rewards, our simulation script will simulate the movements of multiple aircraft arriving at our simple airport. At each time step, there is a random chance that an aircraft (with a randomly assigned airline will land). If this occurs, the aircraft will ask for a gate assignment before taxiing. The chosen gate number represents the action value in the MDP. The file features many constants such as the chance of spawning a new aircraft, the number of gates at the terminal, and the number of possible airlines that can be altered if needed. The constants we chose are shown in Fig. 2.

```
NUM_TAXI_SEGMENTS = 8
NUM_GATES = 8
NUM_AIRLINES = 4

TAXIWAY_LAYOUT = [0] * NUM_TAXI_SEGMENTS
GATE_LAYOUT = [0] * NUM_GATES
GATE_TO_AIRLINE = [1, 2, 2, 3, 3, 3, 4, 4]  # Gates that each airline prefers

MOVE_PROB = 0.8  # move probability -- we can change this
SPAWN_PROB = 0.3  # probability of spawning a new plane each step
```

Fig. 2. Simulation Script Constants

When an aircraft is assigned to a gate, it will move along the taxiway towards that gate. However, there is a chance that the aircraft will not move at full speed, meaning it may remain in place for more than one timestep. When an aircraft arrives at its gate, a reward is assigned. If this gate is preferred by the airline corresponding to that aircraft, the reward will be higher. If an aircraft attempts to move and is blocked by an aircraft ahead or tries to part at a gate that is already occupied, a negative reward or cost is assigned each time this occurs. The values used for these rewards are shown in the following table.

Reward Parameter	Value
Final gate reached with correct airline	+50
Final gate reached with incorrect airline	+10
Airplane attempts to move forward is blocked	-50
Airplane attempts to enter a gate that is already occupied	-100

The state is captured by the occupancy of the taxiway segments, gates, the assigned gate of each aircraft, and the airline of each aircraft. With our chosen constants that means there are 16 total cells with 33 possible values. This means there are 33^{16} distinct states, even for a simple airport! We will discuss how to manage a state space like this in the Solution Methods section. The simulation will end once a certain number of aircraft park at their gates. In our case, that number is three. The script can also detect if there is a blockage preventing more aircraft from parking, which will cause the simulation to end early.

B. Simulation Modes

The simulation script has three modes. The first is designed to generate data for offline models. It stores the state, action, reward, and new state from each step of the simulation. Each state is given a unique integer identifier. In this mode, the gate assignment action is performed randomly over a large number of episodes. Our code is able to run 10,000 full simulations in about 6 seconds. As these simulations are being performed, the MDP data is written to a csv file similar to the data of project 2.

The second mode is designed to interact with online models like MCTS. This mode is similar to the one described above with one major difference. The algorithm for online models like MCTS interacts with the environment through repeated steps of exploration and refinement. Therefore, this mode of simulation can be exported as a step-wise function. In the MCTS model, we only need to call `node_env.step(action)` to take this step.

The final mode is called interactive mode. It allows human players to make gate assignment decisions themselves. This mode will output the current airport environment at every step directly to the terminal. This mode is most useful for familiarizing users with the dynamics of the gate assignment problem and for validating policies.

C. Visualization

One element of the 30-hour extension was creating better terminal visualization logic. In the interactive simulation mode, we added a compact visualization of the airport environment as shown in Fig. 3. An aircraft with a certain airline will be displayed as a string, and will intuitively move along the taxiway and park at a gate. When a reward or cost is incurred, the value and the reason will also be displayed in the terminal. This logic is designed to be robust to multiple aircraft movements without text overflow or lag.

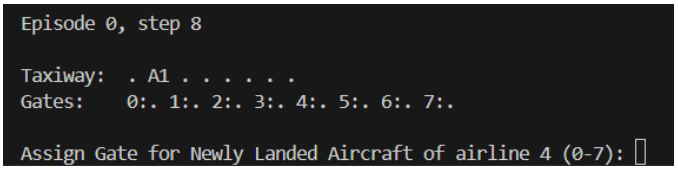

Fig. 3. Terminal Window Airport Visualization

IV. Solution Methods

A. Offline Policies

We first evaluated offline policies using Q-learning, SARSA, and Double Q-learning on a fixed dataset of logged transitions from simulated airport operations. While these methods produced reasonable value estimates within the dataset, they did not perform well when deployed online. Offline learning is limited because it can only learn from the transitions it has already seen and cannot adapt to new aircraft arrivals, different traffic sequences, or changing congestion conditions. As a result, the offline policies often made poor gate assignments and failed to handle blocking situations that were not present in the csv data. Due to these restrictions, we shifted our focus to online methods that simulate future states directly rather than relying on static training data.

B. Online Policies

We tested Monte Carlo and Q-learning as online decision-making policies and compared their performance to a random baseline. For Monte Carlo, the action values were updated using the average return from rollouts (Equation 1), and for Q-learning, the values were updated using the standard temporal-difference update rule (Equation 2). After running repeated simulations, Monte Carlo consistently achieved higher rewards and lower deadlock rates than Q-learning. Because of this consistent performance advantage, we selected Monte Carlo Tree Search (MCTS) as the primary planning method for further improvements (Table 1).

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} G_i \tag{1}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{2}$$

C. Improvement in Online Policies

To improve the accuracy of Monte Carlo value estimates, we modified the rollout policy used during simulated trajectories. In standard Monte Carlo planning, rollouts select future actions randomly, which can produce unnecessary blocking and noisy value estimates. To address this, we introduced a heuristic rollout policy that prefers assigning arriving aircraft to empty gates whenever possible. The function first identifies all empty gates in the simulated environment and selects one at random from this subset. If no empty gates are available, the rollout falls back to a uniform random gate assignment. To prevent the rollouts from becoming overly deterministic, we mixed this heuristic with a small amount of randomness, using a 75% heuristic and 25% random selection probability.

D. Evaluation

We evaluated random assignment, Monte Carlo Tree Search, online Q-learning, and improved MCTS using the same online environment setup. Each policy was tested over 50 episodes, starting from identical initial conditions. For every run, the episode reward, number of steps, completion status, and deadlock outcome were recorded. After running all episodes, the script calculated average reward, completion rate, deadlock rate, and average steps, providing consistent metrics for comparing online decision-making performance across all methods (Table 1).

E. Further Analysis on the Improved MCTS

We tested four main parameters to understand how they influence the performance of the improved MCTS model in the airport environment. First, we examined how the number of rollouts per action influences decision quality (Figure 5). A rollout is a simulated sequence of future interactions starting from the current state, used to estimate the value of a gate assignment. Increasing the number of rollouts allows the agent to evaluate more possible future outcomes before choosing an action, which generally improves decision accuracy but increases computation time. As expected, mean reward improved as rollout count increased, but performance gains became negligible beyond approximately 40 rollouts per action.

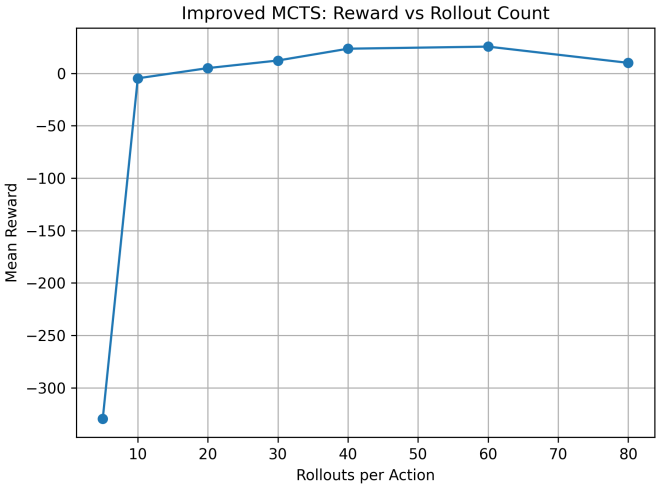

Fig. 4. Mean reward v number of rollouts per action

Second, we tested the heuristic probability, which controls how often rollout simulations follow a congestion-aware rule instead of assigning gates randomly (Figure 6). Higher heuristic probability generally improved mean reward because the rollouts reflected more realistic gate usage patterns. Performance increased up to a certain point, after which additional heuristic weighting provided little benefit and slightly decreased performance at a fully deterministic value of 1.0.

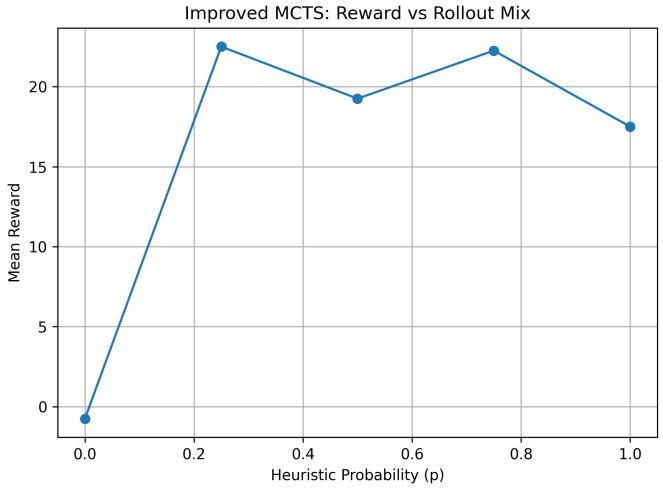

Fig. 5. Mean reward v heuristic probability in improved MCTS

Lower heuristic values introduced more randomness, while higher values made rollouts more structured.

Finally, we tested traffic density by changing the target number of planes required to complete an episode (Figure 7). As expected, mean reward decreased as the number of planes increased, since higher traffic levels create more blocking and gate conflicts. Eventually, this number also surpasses the total number of gate available, which is eight in our case. These results indicate that the algorithm performs well under moderate traffic but may require additional improvements to maintain stability and performance in more congested airport conditions.

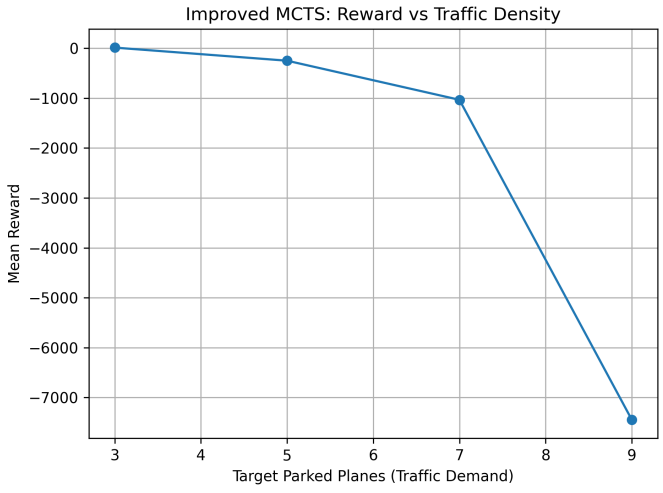

Fig. 6. Reward v traffic density

V. Discussion and Conclusion

Initial experiments with offline models were unsuccessful. Offline agents were trained on a fixed dataset, while realistically, the airport environment evolves dynamically. The encoded state space was also extremely large (33^{16}), making it computationally impractical to estimate actions for all integer-encoded states. These constraints led to unstable value estimates and consistently negative performance, indicating that offline methods are not suitable for this domain unless significant simplifications are made.

TABLE I
Online Policy Performance Comparison

Method	Reward	Completion Rate	Deadlock Rate	Steps
Random Policy	-1456.2	0.86	0.14	24.88
Q-Learning	-461.6	0.94	0.06	19.20
MCTS	-1.0	1.00	0.00	15.32
Improved MCTS	**19.4**	**1.00**	**0.00**	**13.98**

Online models performed substantially better since they evaluate actions directly during simulation. Both Q-learning and MCTS outperformed random gate assignment, with MCTS exhibiting stronger results. To enhance MCTS, we modified the rollout policy to assign aircraft to empty gates during episodes and mixed this heuristic with random actions. This produced more realistic future trajectories and yielded higher rewards than standard MCTS. Since there is a cost accrued each time step when an aircraft is blocked, the random policy can quickly make costly mistakes. As shown in Table 1, the average reward of the random policy is -1456.2. Improved MCTS obtained an average reward of 19.4 indicating a significant improvement in problem-solving ability.

Further analysis showed that performance generally increased with rollout count and heuristic mixing probability. However, reward decreased under higher traffic density, indicating that additional modeling is necessary for congested airports.

VI. Contributions

Both authors contributed equally to performing experiments and writing this paper. Vale focused on literature review and simulation of the problem. Rana developed the solution methods and validation of policies. Following this distribution of work, Vale wrote the Abstract, Introduction, Simulation, and Future Work section. Rana wrote the Solution Methods and the Discussion and Conclusion. Each other recieved feedback and revisions from the other.

VII. Future Work

Our project has demonstrated the great potential in using MDPs to model airport gate assignment. There are a variety of future directions that could be investigated. Given our time constraints, we were only able to gather data and validate our models using a simulation of a simple airport. In the real world, taxiway and terminal layouts are unique and complex. This adds an additional challenge to the AGAP. It would be interesting to see if the MCTS solution method would still provide a benefit for these airport layouts. There are also additional factors that could be introduced for more informed optimization. For example, some gates might be able to handle large aircraft while others cannot. These factors and their

associated rewards could be readily implemented with our MDP strategy.

Gathering real-world data from sources such as FlightAware and ADSB Exchange would also be an interesting next step. This data could help validate our simulated values and assumptions. Overall, we believe that the MDP method is a promising approach to solving AGAP, and we recommend future scholars investigate its potential further.

REFERENCES

[1] Aoun, O., & El Afia, A. (2014, June). Using Markov decision processes to solve stochastic gate assignment problem. In 2014 International Conference on Logistics Operations Management (pp. 42-47). IEEE..

[2] Bouras, A., Ghaleb, M. A., Suryahatmaja, U. S., & Salem, A. M. (2014). The airport gate assignment problem: a survey. The scientific world journal, 2014(1), 923859.

[3] Kochenderfer, M. J., Wheeler, T. A., & Wray, K. H. (2022). Algorithms for decision making. MIT press.

[4] Rome, J. A., Rose, S. D., Lee, R. W., Cistone, J. H., Bell, G. F., & Leber, W. S. (2001). Ripple delay and its mitigation. Air Traffic Control Quarterly, 9(2), 59-98.