Autonomous Approach ATC with Partially Observable Aircraft Positions

Vignesh Anand, Sidharth Tadeparti, Sneha Jayaganthan

December 2023

***Abstract*—Air traffic control capacity shortages are the leading cause of flight delays in Europe. This project addresses the problem of terminal approach control for arriving aircraft around an aerodrome. The Problem considers N cooperating aircraft in a discretized airspace 2-D with uncertain location information and compliance errors. The problem is formulated with and without the assumption of full observability and solved using online methods tree search methods with information from offline training. The resulting solvers are able to navigate aircraft to the airfield while maintaining adequate separation safely.**

***Code Repository*: github.com/sidt36/AA228_Project_Final**

Introduction

Air Traffic Control (ATC) services are crucial in enabling safe and efficient traffic flow in modern airspaces. Today, ATC is a manpower-intensive operation with a large number of complexities involved. With the ongoing controller shortages around the world [1] contributing to 1000's of delays daily, it is prudent to explore autonomous approaches to ATC. In particular, we choose to tackle the problem of approach control, which is responsible for providing ATC services to arriving traffic around an aerodrome [2]. We discretize the 2-D airspace as a 30x30 occupancy grid and account for uncertainties in aircraft compliance and errors in the observation of position.

I. Related Work

The topic of aereal collision avoidance has been extensively studied [3], however the automating ATC itself hasn't been as popularly studied. Previous work in [4] explores applying AI techniques to ATC automation, specifically developing a computer program that automates rudimentary air traffic control planning and decision-making functions. More relevantly, [5] proposes modelling air traffic collision as a continuous time Markov Decision Process (CTMDP). A more advanced deep multi agent RL approach is explored by [6] which uses an actor critic method to realize policy gradient. The model used has discrete state and action space. The authors also achieve autonomous vectoring and sequencing in [7] using hierarchical RL. In this project we take inspiration from previous work, and utilize a discretized state and action space in MDP framework. While the previous investigations have been dealt with the problem of Autonomous ATC using sophisticated approaches, they don't rigorously consider partial observability. This further inspires us to model the problem as a POMDP as well to understand the impact of partial observability.

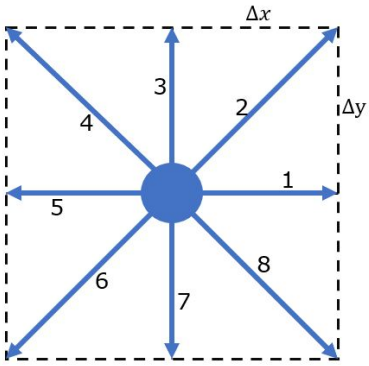

Figure 1: The eight neighbors of a given state.

II. Problem Description

A. State and Action Space

The problem features N cooperating aircraft occupying a few discretized positions. A representative 30nm $\times$30nm airspace around the active runway, discretized into 1nm $\times$1nm grid, this is depicted in figure 2. We also account for a single terminal state representing all configurations outside the airspace. This gives us a $2N$ dimensional state vector with a state space of 901^n. The ATC algorithm would be able to command each aircraft to move one of the eight neighboring positions. This gives us an action space of size 8^n.

B. Objective and Uncertainty

The goal of the designed policy is to

- Maintain safe separation between all aircraft in the airspace
- Navigate aircraft to the goal state (airfield)
- Avoid navigating aircraft to positions outside the airspace

The sources of uncertainty encoded in our decision-making problem include:

- Uncertain or incomplete information of aircraft states.
- Uncertain errors in complying with ATC instructions.

We chose to solve four different decision-making problems.

- **MDP 1:** Single aircraft with uncertain compliance but full observability.
- **MDP 2:** Two aircraft with uncertain compliance but full observability.
- **POMDP 1:** Single aircraft with uncertain compliance and partial observability.

- **POMDP 2:** Two aircraft with uncertain compliance and partial observability.

C. Transition and Observation Model

The problem state is represented as a tuple of x and y coordinates of the form,

$$\mathbb{X}^n = (x_1, y_1, x_2, y_2, ...x_n, y_n),$$

where, $x_i, y_i \in 1 \cdot 30$. Additionally, if an aircraft leaves the confines of the described airspace, it enters the terminal state given by $(-1, 1)$.

Actions were encoded by a scalar integer $a \in [1, 8]$ for each aircraft representing the eight neighbors of a grid spot; this is depicted in figure 1. The action for the multi-aircraft scenario is encoded as a tuple,

$$\mathbb{A}^n = (a_1, a_2, ...a_n).$$

The transition probability of entering the commanded state is 0.8, with 0.1 probability of reaching each of the two adjacent states. An aircraft in a goal state remains in the goal state with full probability. For cases of partial observability, our observation model is given by

$$O(s) = \begin{cases} s & p = 0.8 \\ \text{Neighbour}(s) & p = 0.2 \end{cases} \tag{1}$$

Where Neighbour(s) randomly samples a position from one of the neighbors of the current state.

III. Problem Approach

A. Reward Landscape

To achieve the goals stated in 2, the reward landscape was designed as follows

- $100e^{-d/3}, \quad d < 5$: Where d is the distance from the goal runway state. This serves as a reward to aircrafts that come in for landing.
- -100: Leaving the problem domain and entering a terminal state that represents states outside the problem domain. This penalty is assessed for each aircraft that leaves the designated airspace.
- $-150e^{d_{ij}/7}, \quad d_{ij} < 5$: Where d_{ij} is the distance between adjacent aircrafts that haven't already landed. This term serves as a penalty for aircrafts that come close to each other.

B. Solution Methods

To solve our MDP/POMDP problems, we use a variety of solution methods found in [8] and [9] as detailed below:

Discrete Value Iteration: Discrete Value Iteration is an iterative scheme to solve MDPs exactly. An MDP can be solved by beginning with a bounded value function U and applying the bellman update given as follows:

$$U_{k+1}(s) = max_a \left(R(s, a) + \gamma \sum_{s'} T(s'|s, a) U_k(s') \right) \tag{2}$$

Neural Network Approximate Value Iteration: In this variation of value iteration for MDPs, an approximate neural network representation of the value function is maintained through the means of a neural network. At each iterative step, n_s random states are sampled, and one bellman update (Eq. 2) is performed using approximate values for utility. The neural network representation is then retrained on the updated estimates of utility.

QMDP: QMDP involves iteratively updating a set τ of α-vectors. The resulting alpha vectors define a value function and policy from a belief state. Each α-vector is initialized to zero and is updated as follows:

$$\alpha_a^{k+1}(s) = R(s, a) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} \alpha_{a'}^k(s) \tag{3}$$

Monte Carlo Tree Search (MCTS): The MCTS solver executes m simulates from the current state. The algorithm updates the estimates of action value function $Q(s, a)$ and $N(s, a)$. At each state, actions are chosen by maximizing the UCB1 heuristic given as follows:

$$Q(s, a) + c\sqrt{\frac{\log N(s)}{N(s, a)}} \tag{4}$$

Utility estimates at the lowest levels are supplied using offline methods outlined in 3.2,3.2.

AR-DESPOT: Anytime Regularized Determinized Sparse Observable Tree Search is an online Solver that utilizes a set of independent bounds to solve a POMDP. While a standard belief tree captures the execution of all policies under all possible scenarios, a DESPOT captures the execution of all policies under a set of sampled scenarios [10]. AR-DESPOT extends this idea by incorporating ideas of regularization to avoid overfitting and pruning to improve accuracy and speed.

IV. Analysis and Results:

This section details the implementation, the results, and a subsequent analysis of the results. The following results were obtained with MDP/POMDP models defined and solved with POMDPS.jl module [9]. The performance of the proposed methodology is examined against a random policy and is also visualized. The aircraft positions at various times are represented using circular markers that are blue and red in color.

A. MDP - 1 Aircraft

For the case of the single-aircraft, we obtain a reasonable state space size of 901, comprising 900 grid spots and one terminal state representing all configurations outside the grid. The action space size was also small, with 8 possible actions at each state.

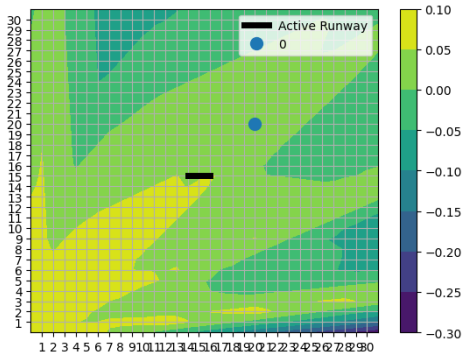

Figure 2: Value Estimates with approximate value iteration

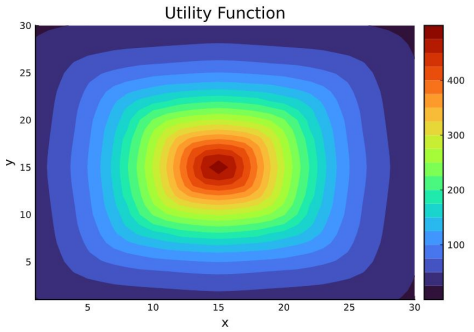

Figure 3: MDP-1 aircraft state utility contours from value iteration.

Our first attempt was to solve this problem with an Approximate neural value iteration presented in 3.2. A simple neural network was developed using Keras with a model structure given by:

- Input layer: $2 \cdot n$ inputs, where n is the number of
- Hidden Layers: 3 layers of 16 nodes with RELU activation
- Output Layer: Scalar linear output layer

at each iteration, 100 states were randomly sampled for the approximate value iteration update. However, as illustrated in Fig2, no convergence was seen, and values could not be estimated. Later, Direct value iteration was attempted, which accurately captured the utilities of states in the MDP. These utilities are presented in Fig 3. The policy was obtained by acting greedily with respect to the converged values tested using Monte Carlo simulations. Results are presented in Table 1. Our value-based policy outperforms a random policy, which, on average, is able to accumulate no rewards. A representative trajectory is presented Fig 4

Process	Random Policy	Value Iteration Policy
MDP-1 Aircraft	0	40.6334

Table 1: Results - Random Policy vs MDP (1 Aircraft)

B. MDP - 2 Aircraft

The state and action space grows exponentially with the number of aircraft. The state and action spaces for a 2-aircraft MDP are of size $(901)^2$ and $(8)^2$, respectively, which represents the

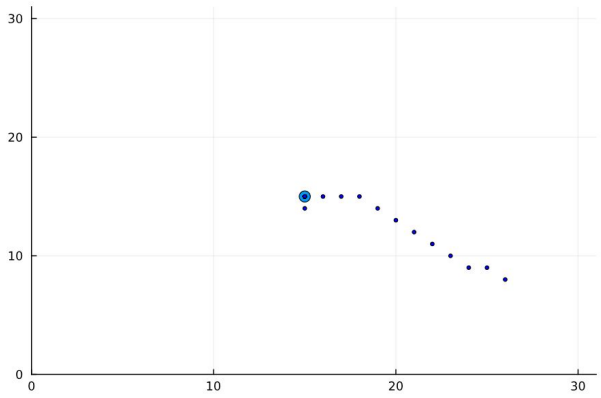

Figure 4: MDP-1 Trajectory

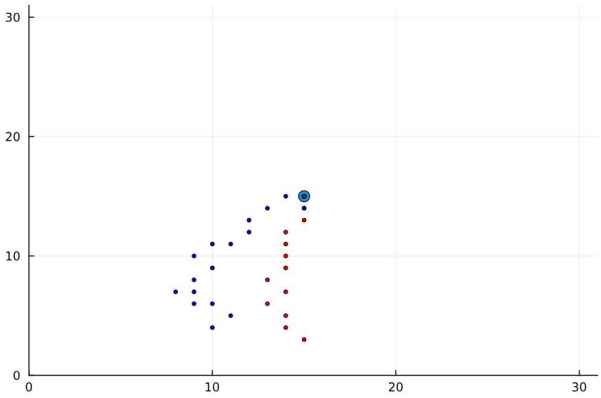

Figure 5: MDP - 2 Aircraft

size limit for MDP an explicit MDP representation on standard laptop hardware. Large state space meant that this problem could not be solved using exact or approximate value iteration and necessitated the use of an online method. The chosen online method was the Monte Carlo Tree Search from 3.2. While the MCTS solver was able to capture penalties associated with leaving the airspace or collisions with other aircraft, it often failed to capture the reward associated with the runway state. This necessitated providing additional utility estimates from the previously solved one-aircraft problem in 4.1. The utility of a combined aircraft position state represented as $(s1, s2, s3, s4)$ is estimated as $\sum_s U_{VI}(s)$ where U_{VI} represents the optimal utility estimate for the single aircraft case. The utility estimates inform the MCTS solver of potential future rewards associated with runway proximity, while the online planning helps avoid penalties from collision or exiting the airspace. A representative trajectory arising out of this policy is presented in Fig 5. Results from Monte Carlo analysis for the online policy and comparisons with a random policy are presented in Table 2. We find that, on average, a random policy accrues penalties associated with proximity to other aircraft and airspace breaches

C. POMDP - 1 Aircraft

The MDP formulations discussed previously successfully guide the aircrafts to the runway without collision. However,

Process	Random Policy	Augmented MCTS
MDP-2 Aircraft	-154.286	82.7082

Table 2: Results - Random Policy vs MDP (2 aircraft)

the assumption of complete observability of the state is a strong assumption to make. In this section the MDP is converted into a POMDP using the observation model previously described. The beliefs of the states are updated using a discrete Bayes filter given the discrete nature of the problem.

The single aircraft case of the POMDP is solved using QMDP, which gives us an upper bound on the optimal value function and alpha vectors. The single aircraft problem consists of 901 states and hence an online method such as QMDP is tenable. As in the previous case, the policy obtained from the QMDP algorithm is evaluated against a random policy. As in the previous cases, the sum of discounted rewards to a depth of 30 is computed using the parallel simulator from the POMDPs.jl package.

Process	Random Policy	QMDP Policy
POMDP-1 Aircraft	-958.143	73.3886

Table 3: Results - Random Policy vs POMDP (1 aircraft)

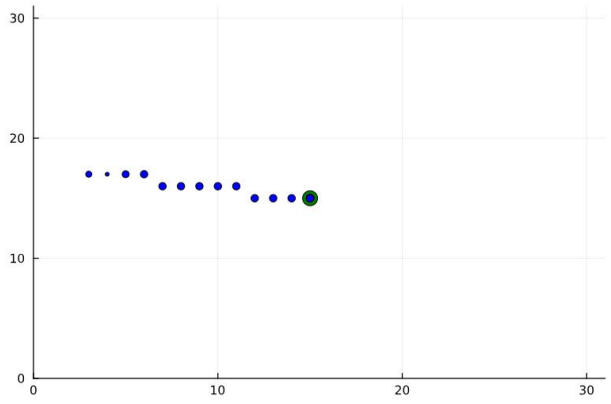

Figure 6: POMDP - 1 Aircraft

The QMDP policy clearly outperforms the random policy which incurs severe negative reward during the simulations. In a single aircraft case a negative penalty is incurred when the aircraft leaves the bounds of the airspace, which occurs often with the random policy.

The QMDP policy is visualized in figure 6. Unlike in previous cases, where we knew exactly where the aircrafts were, here we only have a belief over the possible states of the aircrafts. This has been represented by augmenting the size proportional to the magnitude of the belief associated with that point in the trajectory.

We can observed that while the belief over the state is initially weak, within a couple of points it improves to near certainty with the help of the observation model.

D. POMDP - 2 Aircraft

The final problem involves two cooperative aircrafts that are partially observable. As in the case of 2 aircraft MDP, an offline method such as QMDP isn't feasible with limited computational resource. Therefore AR-DESPOT, an online method is used to compute policies.

A key limitation in the case of POMDPs is the inability to pass on utililty estimates to AR-DESPOT. This result's in the online method having to shoot in the dark with a limited visibility of the reward landscape especially at locations that far away form the runway.

This results in abysmal performance by the AR-DESPOT solver primarily due to weak upper and lower bounds provided. In the MDP case, the complete knowledge of the value at a given state was leveraged to inform MCTS better. A similar approach with QMDP was found to be futile, which is likely due the estimate not translating well in the two aircraft case.

Process	Random Policy	AR-DESPOT Policy
POMDP-2 Aircraft	360.552	393.187

Table 4: Results - Random Policy vs POMDP (2 aircraft)

The performance of the AR-DESPOT policy can be seen in figure 7, where both the aircrafts oscillate without moving in constructive directions .This in stark contrast to the MDP case where both aircrafts successfully landed while maintaining a safe distance from each other. This highlights the complexity that partial observability induces in problem with large state spaces.

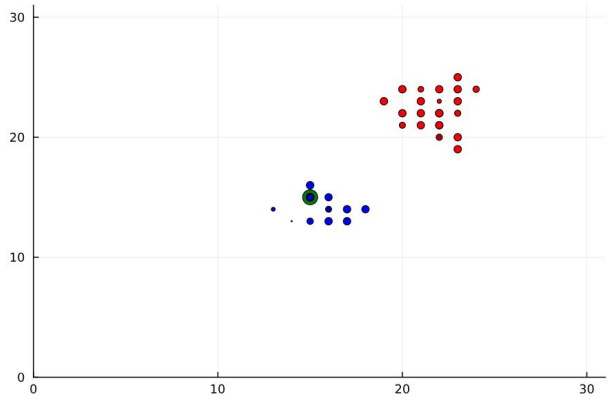

Figure 7: POMDP - 2 Aircraft

V. Conclusion

This project formulated and solved one and multi-aircraft decision-making problems for air traffic control. For the outlined MDP problems, a combination of online planning and offline estimates was able to deliver high-performance and safe policies. For POMDP problems, while single aircraft problems were easily solved with offline methods, the online AR-DESPOT solver was not able to significantly outperform a random policy due to poor quality estimates on utility bounds.

VI. Future Work

- **High Fidelity State Representations**: Solve MDPs and POMDPs with finer or continuous representation of aircraft position, with models for airspeed and altitude
- **Advanced Pilot Models**: Build more advanced pilot models for better representation of transition dynamics.
- **Uncooperative Aircraft**: Develop policies to identify malicious or uncooperative aircraft and safely navigate air traffic.

VII. Contributions

- **Author 1 (Sidharth, sidt):** Implementation of MDP Model Definition in Python and Julia (Rewards, Transition, State and Action Space), observation model, GUI Rendering, MONTE Carlo Evaluation
- **Author 2 (Vignesh, viganand):** Conceptualization of Problem, Mathematical Descriptions, Putting the components together - POMDPs.jl structuring, solver interfacing, quickPOMDP/MDP. Interpretation of Results.
- **Author 3 (Sneha, snehajay):** Python Implementation of MDP Model Definition, Approximate Value Iteration, POMDP and MDP performance evaluation and compilation, NN setup, literature review, report formatting.

References

[1] D. Shepardson, "Faa must address us air traffic staffing crunch, nominee says," Oct 2023.

[2] S. A. Safety, "The approach controller."

[3] I. Mahjri, A. Dhraief, and A. Belghith, "A review on collision avoidance systems for unmanned aerial vehicles," in *Communication Technologies for Vehicles* (M. Kassab, M. Berbineau, A. Vinel, M. Jonsson, F. Garcia, and J. Soler, eds.), (Cham), pp. 203–214, Springer International Publishing, 2015.

[4] D. A. Spencer, "Applying artificial intelligence techniques to air traffic control automation,"

[5] M. K. Zouhair Mahboubi, "Continuous time autonomous air traffic control for non-towered airports,"

[6] M. Brittain and P. Wei, "Autonomous air traffic controller: A deep multi-agent reinforcement learning approach," 2019.

[7] M. Brittain and P. Wei, "Autonomous aircraft sequencing and separation with hierarchical deep reinforcement learning," 06 2018.

[8] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for decision making*. London, England: MIT Press, Aug. 2022.

[9] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, "POMDPs.jl: A framework for sequential decision making under uncertainty," *Journal of Machine Learning Research*, vol. 18, no. 26, pp. 1–5, 2017.

[10] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *Journal of Artificial Intelligence Research*, vol. 58, p. 231–266, Jan. 2017.