Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

A MULTI-AGENT REINFORCEMENT LEARNING APPROACH FOR SYSTEM-LEVEL FLIGHT DELAY ABSORPTION

Kanupriya Malhotra
Zhi Jun Lim
Sameer Alam

Air Traffic Management Research Institute
Nanyang Technological University
65 Nanyang Drive
Singapore, 637460, SINGAPORE

ABSTRACT

With increasing air traffic, there is an ever-growing need for Air Traffic Controllers (ATCO) to efficiently manage traffic and congestion. Congestion often leads to increased delays in the Terminal Maneuvering Area (TMA), causing large amounts of fuel burn and detrimental environmental impacts. Approaches such as the Extended Arrival Manager (E-AMAN) propose solutions to absorb such delays, whereby flights are scheduled much before they enter the TMA. However, such an approach requires a speed management system where flights can coordinate to absorb system-level delays in their en-route phase. This paper proposes a Multi-Agent System (MAS) approach using Deep Reinforcement Learning to model and train flights as agents which can coordinate with each other to effectively absorb system-level delays. The simulations utilize Multi-Agent POsthumous Credit Assignment in Unity and test two reward approaches. Initial findings reveal an average of 3.3 minutes of system-level delay absorptions from a required delay of 4 minutes.

1 INTRODUCTION

According to the International Civil Aviation Organization (ICAO), the total number of passengers carried by service airplanes increased to 4.5 billion in 2019, which marked a 3.9 percent increase in air traffic as compared to 2018 (International Civil Aviation Organization 2019). While the COVID-19 pandemic resulted in a downfall in air traffic in the year 2020, air traffic growth is predicted to continually rise in the coming years, as the world recovers from the pandemic (Airports Council International 2021). Due to the continued increase in air traffic, techniques to monitor and decrease congestion, especially during flight arrival stages, are being thoroughly explored. An increase in congestion essentially results in delays in the Terminal Maneuvering Area (TMA), causing enormous fuel burn and longer holding periods (Ma et al. 2019). Various solutions have been proposed over time to address these concerns, with one of the prominent ones suggesting the transfer of TMA delays to the cruise phase of the flight. The Extended Arrival Management, E-AMAN (SESAR 2022), developed by the Single European Sky ATM Research (SESAR) Program, proposes one such solution to effectively absorb TMA delays, and transfer these to the cruise or en-route phase of the flight. The E-AMAN approach focuses around absorbing these delays by aiding ATC to sequence traffic much before they enter into the TMA. The initial findings from the implementation of the E-AMAN approach in the London TMA/Heathrow Airspace exhibit that reducing 1 minute in the TMA could potentially yield a reduction in CO_2 emissions by approximately 5,000 tonnes. Moreover, this could also contribute to a yearly savings of €1.25 million (SESAR 2015). The extent to which this approach can contribute to combat losses can be further highlighted by the potential reduction in fuel burn by 60kg per minute reduction in the holding period. While implementing the E-AMAN

978-1-6654-7661-4/22/$31.00 ©2022 IEEE

approach poses numerous benefits, and various prediction models exist to predict flight Estimated Time of Arrival (ETA) and delays, an efficient speed control strategy is required for practical implementation of the technique. This mechanism requires a system that allows for the interaction between multiple flights in their cruise phase, such that flights can coordinate regarding TMA delay and execute speed management strategies on a system-level bases.

2 BACKGROUND

Multiple strategies have been proposed in previous researches to mitigate and absorb the delay in the TMA during arrival. Past researches by Carlier et al. proposed holding stack and possible flight re-routing to combat negative environmental impacts due to long arrival TMA delays (Carlier et al. 2007). However, increase in transit time often results in greater fuel burn and does not reduce TMA congestion. Thus, speed control strategies to combat delays in the airspace were encouraged to provide the most feasible results.

Another approach implemented by Delgado et al. conducts a cruise phase/en-route speed reduction to accompany current practices in ATFM, by analyzing the range of speed reduction that can be accommodated while a flight is cruising with a similar or lower fuel consumption (Delgado and Prats 2012). It further focuses on the impact of altitude on the airborne delay of the flight. This range was analyzed to fall within 5-12% of the flight's original speed. Research conducted by Yoshinori et al. focused on speed control methods to achieve airborne delays (within the range of 2-6 mins) in the Tokyo International Airport, by altering the compliance rates of the flights (Matsuno et al. 2020). It was presented that speed reduction could take place for a range of 2 – 3 minutes in 30-minute intervals. Recent work by Dhief et al. involved improving the speed control methods by coupling them with accurate TMA delay prediction models in the Singapore TMA (Dhief et al. 2020). Holding prediction models were implemented at a 100 NM interval in the ranges 300-500NM to evaluate whether a flight would enter a holding pattern, and implement delay prediction models coupled with speed control management to absorb maximum possible delay.

While these researches support that speed control management is an effective method to combat TMA congestion and successfully transfer TMA delays to the cruise phase, most speed control management largely takes place on an individual flight basis. A flight's ability to absorb delay, in most theoretical cases, solely impacts the results of the individual flight. However, in practical implementation, it must be ensured that all flights are working towards maximizing system level delay absorption, and that delay absorption is fair so that no flight is greatly penalized. Thus, the strategy should accommodate a multi-flight interaction, where the success or failure of a flight to absorb delay is reflected on other flights so that the system aims to maximize the total delay absorption, while aiming to distribute the delay equally.

This report proposes a cooperative Multi-Agent System (MAS) approach, where flights are simulated as agents that can coordinate to absorb maximum possible system-level delays by speed modifications in their cruise phase. In the multi-agent system, each flight is conveyed with information regarding its spatial positions in the environment, along with its Scheduled flight path and Scheduled Estimated Time of Arrival (ETA). The multi-agent system is provided delays on an individual level and is trained to absorb system-level delays using Deep Reinforcement Learning Techniques. The goal of each flight is to absorb the maximum possible delay that has been assigned to it by speed modifications, while avoiding any possible collisions with other flights due to speed modifications.

The MAS' flight paths and movements have been simulated using the Unity 3D Platform, and Deep Reinforcement Learning has been applied using the Unity ML-Agent Toolkit. The system is trained by providing rewards to all agents, based on the Actual Arrival Time of the Flight. Rewards are provided on a MAS group basis, to ensure that the actions of one flight have an impact on all flights, and that one flight is not penalized more than another. The training of such a model can aid in collecting speed modifications during the course of the flight plan, and instantiate another flight plan that would provide recommended flight speeds for the maximum possible delay absorption in the cruise phase. Figure 1 shows the overview of the TMA delay transfer to the cruise phase of the flight.

The organization of the paper is as follows. Section 3 discusses the requirements for the MAS, along with the proposed framework for the agents and the MAS. Section 4 discusses the simulation and experimentation of the Multi Agent System on Unity. Section 5 highlights the results obtained from training the Reinforcement Learning (RL) models using MA-POCA, and Section 6 concludes the report.

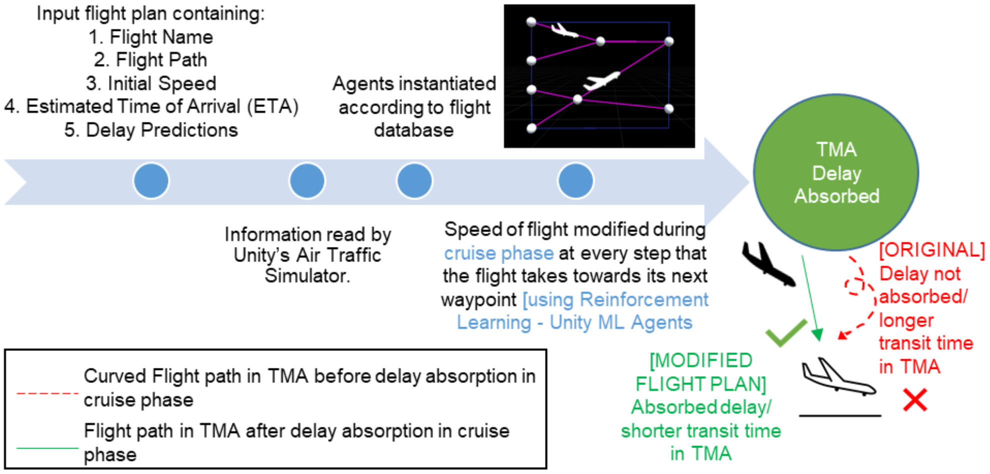

Input flight plan containing:
1. Flight Name
2. Flight Path
3. Initial Speed
4. Estimated Time of Arrival (ETA)
5. Delay Predictions
Agents instantiated according to flight database
Information read by Unity's Air Traffic Simulator.
Speed of flight modified during cruise phase at every step that the flight takes towards its next waypoint [using Reinforcement Learning - Unity ML Agents
TMA Delay Absorbed
[ORIGINAL] Delay not absorbed/ longer transit time in TMA
[MODIFIED FLIGHT PLAN] Absorbed delay/ shorter transit time in TMA
Curved Flight path in TMA before delay absorption in cruise phase
Flight path in TMA after delay absorption in cruise phase

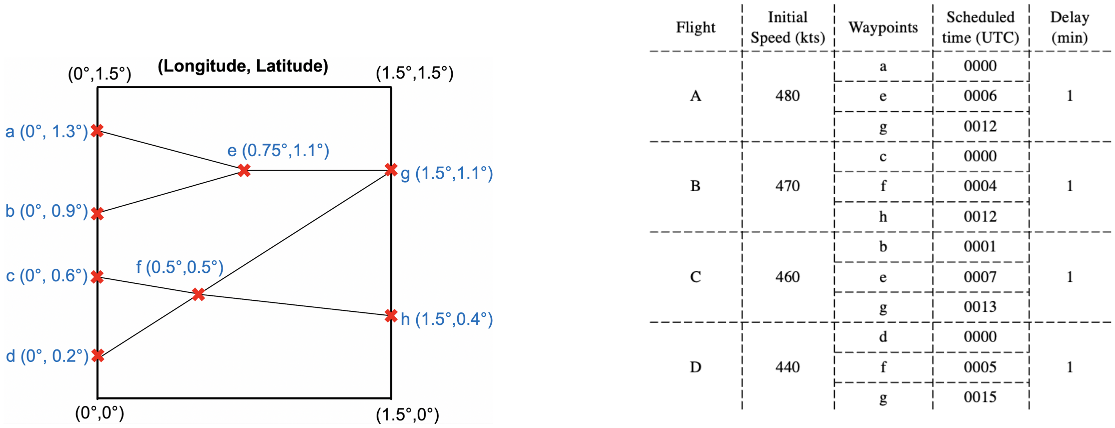
Figure 1: Concept diagram of TMA delay absorption in the cruise phase.

3 METHODOLOGY

3.1 Problem Formulation

To develop a multi-agent system, a sample flight plan for the cruise phase of multiple flights was constructed, as highlighted in Figure 2. An airspace consisting of 4 flights in their cruise-phase was modelled, along with the waypoints that each flight was required to travel through. The simulation environment contained Scheduled ETAs for each flight's destinations, along with the required delay that each flight aimed to absorb. The flight plan consisted of 8 waypoints in total, with each flight designated to travel to 3 waypoints. The flight delay value provided in the flight plan contained the maximum cruise phase delay that the flight attempted to absorb during its course in the simulation. Each of the 4 flights were provided with a required delay time of 1 minute, resulting in a total desired system-level delay of 4 minutes. It must be noted that all flights were originally travelling at constant deceleration between the two waypoints. Each flight was also assigned with an initial speed, which refers to the speed of the flight at the first waypoint.

The goal of the simulation was to use speed regularization to maximize the system-level or cumulative delay absorption that had been provided to the flights, such that the delay absorption is performed in the most distributed manner possible. Each flight was treated as an agent, that collected observations about its environment, and performed actions at each step. These actions included speed modifications to the original speeds of the flight by ±10% only at waypoints, as analyzed by the literature, which highlighted that a cruise speed reduction by 5-12% would permit flights to fly with similar or lower fuel consumption, thus making 10% a reasonable limit for the flight's cruise speed reduction (Delgado and Prats 2012). Based on the actions of a single agent, rewards were provided to the multi-agent group, such that a single agent's actions impacted the multi-agent group as a whole. These rewards were beneficial in training policies that defined the behavior of the agents. The flights were penalized if the total delay absorbed by the system exceeded the required delay. To generate flight plans with moderated speeds, a simulation environment was built using the Unity 3D platform. The main requirements of the MAS Simulation based on this problem formulation have been summarized in Figure 3. Since this study deals with flights in their cruise phase, it was assumed that flights travel at constant acceleration between two waypoints.

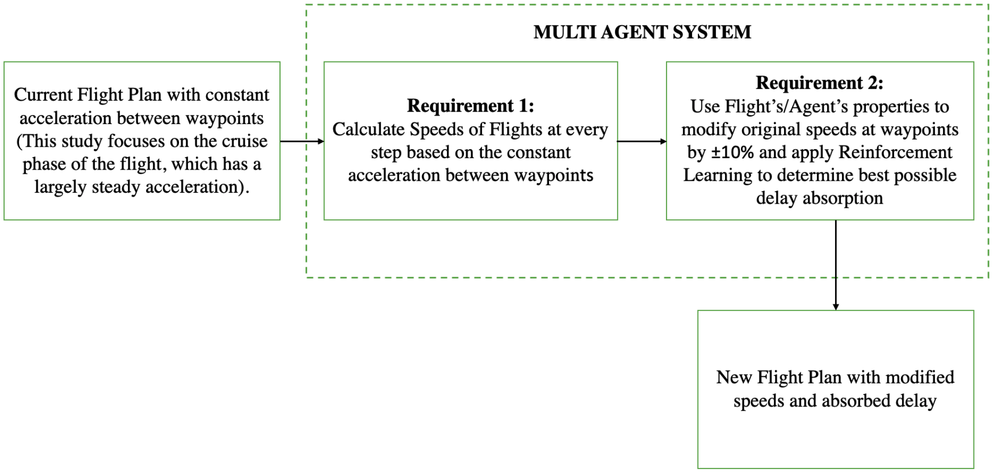


(a) Airspace boundary and waypoints.

Flight	Initial Speed (kts)	Waypoints	Scheduled time (UTC)	Delay (min)
A	480	a	0000	1
		e	0006	
		g	0012	
B	470	c	0000	1
		f	0004	
		h	0012	
C	460	b	0001	1
		e	0007	
		g	0013	
D	440	d	0000	1
		f	0005	
		g	0015	

(b) Sample flight plan input to the simulator.

Figure 2: Sample flight plan for cruise phase simulation. Panel (a) depicts the airspace boundary, where the waypoints have been labelled with their names and positions. Panel (b) highlights the flight plan input into the software, containing the flight's name, its initial speed at the first waypoint, the scheduled time of arrival at each waypoint, and the delay absorption required by each flight in the simulation.


MULTI AGENT SYSTEM

Current Flight Plan with constant acceleration between waypoints (This study focuses on the cruise phase of the flight, which has a largely steady acceleration).

Requirement 1: Calculate Speeds of Flights at every step based on the constant acceleration between waypoints

Requirement 2: Use Flight's/Agent's properties to modify original speeds at waypoints by ±10% and apply Reinforcement Learning to determine best possible delay absorption

New Flight Plan with modified speeds and absorbed delay

Figure 3: Primary requirements of multi-agent system.

3.2 Proposed Framework of Agents

To represent flights as agents, the agents were provided with components of the Markov Decision Process (MDP). The Markov Decision Process is a framework that is often utilized for decision making processes that involve uncertainty (Steimle et al. 2021). The Markov Decision Process focuses on using Markov Chains and uses actions to determine and control rewards and the transitional dynamics of a system. A Markov Chain can be posed as a mathematical framework that adheres to probabilistic rules which determine the transitional changes from one state to the next (Brilliant.org 2022). The Markov Property generally refers to the assumption that the effects of the action taken in a state depend solely on the state and not on any previous states. The RL problem and approach can be posed as a Markov Decision process, which consists of the following components - A set of States (S), A set of Actions (A), Transition dynamics (P), Rewards (R), Discount factor (*gamma*). While the aim of the MAS was to facilitate interaction between the agents and the consequences of their actions, the components of the MDP were first determined on an agent-level basis. This section proposes a framework for the components of an individual flight modelled as an MDP agent, which define the behaviors and learning of the flight during the RL process.

3.2.1 State of the Agent

For this approach, the state of an agent is represented by two variables – the position of the agent in the local airspace, and the position of the next waypoint that the agent is designated to reach. These states were selected due to their ability to aid agents in choosing relevant actions. These two variables are represented by two vectors normalised in the range [-1,1], which allow the agent to gain perspective of its current position relative to its target waypoint. The normalisation was done by dividing the actual vectors by the maximum value in the airspace boundary. For instance, in an airspace boundary of 1.5X1.5, the vector (0.75,1.1) would be normalised to (0.5,0.73). The state of an agent has been further described in Figure 4.

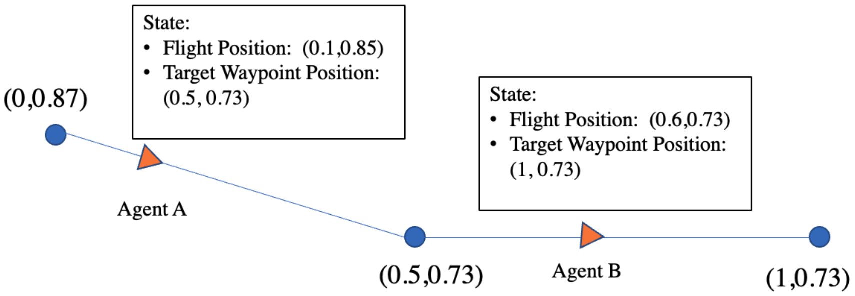

State:
• Flight Position: (0.1,0.85)
• Target Waypoint Position: (0.5, 0.73)
(0,0.87)
State:
• Flight Position: (0.6,0.73)
• Target Waypoint Position: (1, 0.73)
Agent A
(0.5,0.73)
Agent B
(1,0.73)

Figure 4: Sample state of two agents. The state of an agent is defined by two vectors - the position of the agent, and the position of its next target waypoint. This figure highlights normalised vectors that define the states of two distinct agents.

3.2.2 Actions assigned to Agent

The actions provided to the flight include a ±10% moderation of the flight's original speed every time it reaches a waypoint. The moderated speed of the flight effectively results in the change in state, or local position, of the flight. This action was taken by the flight at every waypoint that it travelled to. For this, the original speed of the flight was calculated at every step. Obtaining the original speed of the flight at every step comprised of three main steps:

1. **Calculating the constant acceleration of the flight between two waypoints:**
   The agent travels at constant acceleration between each link. When an agent reaches its target waypoint, the constant acceleration required for the agent to travel to the next waypoint is updated. Thus, the acceleration, a, of the flight travelling from waypoint 1, $w1$, to waypoint 2, $w2$, can be calculated using the kinematics equations for constant acceleration.

$$s = ut + \frac{at^2}{2}$$

   Thus,

$$a = \frac{2}{t}(\frac{s}{t} - u)$$

   where a = constant acceleration between two waypoints (m/s^2); t = time required to reach the next waypoint (s); u = initial speed of the flight at waypoint w1 (m/s); s = distance between $w1$ and $w2$ (m/s)

2. **Calculating the current speed of the flight at every frame**
   Based on the constant acceleration that the agent is traveling at, the kinematics equations utilized to calculate the speed of the agent at every step is as follows:

   $$v = u + at$$

   where u = speed of the flight in the previous frame (m/s); v = speed of the flight in the current frame (m/s); a = constant acceleration of the flight (m/s^2); t = time difference between the two frames (s) Thus, the actions taken by the agent involve modification of the speed,v, only at the step where the agent reaches a waypoint. This speed of the flight, v, at the waypoint, is directly subject to a ±10% change to facilitate delay absorption. This is the Action Implementation Step, as the speed modification action is executed by the flight only when it reaches a new waypoint.
3. **Action Implementation Step - Modifying the calculated current speed by 10%:**
   Actions can fall under two main categories – Continuous and Discrete. Since the agent's action entails it to modify the speed of the flight when it reaches a waypoint within any continuous value lying in the range ±10%, continuous actions were implemented. Since the only direct action was speed modification, the continuous action size for this function was set to 1. After the speed modification was completed at a waypoint, a new constant acceleration was calculated by the simulator for the flight's path to the next target waypoint. Continuous Actions are clamped within the range of [-1,1]. In this case, upon the request of receiving continuous actions, the actions were further clamped to the range [-0.1,0.1]. The formula used to compute the modified speed of the agent at a particular frame, given the original speed, v_o, and a continuous action value, α, was as follows:

   $$v_f = v_o + \alpha v_o$$

   where: v_f = modified speed (m/s); v_o = original speed (m/s); α ε [-0.1,0.1]. After the speed was modified, the agent utilized the modified speed to take its next step towards the goal waypoint, using the following equation:

   $$x_f = x_i + v_f \Delta t$$

   where: x_f = position of the flight in the next frame ; x = position of the flight in the current frame ; v_f = modified speed (m/s); Δt = time difference between two frames (s).

3.2.3 Rewards

Reward is provided to the MAS based on three variables provided to the agent controller by the agent – the Scheduled ETA, the Actual Time of Arrival, and the Required Time of Arrival (Scheduled ETA + required delay). Two approaches have been formulated for the reward structure of the flights.

Approach 1: In this approach, the delay assigned to the flight depends on the individual delay provided by the flight plan. The reward structure is proportional to the delay absorbed by the system. In the case that the flight's delay absorbed exceeds the required delay, the flight is penalised by a very large value of -50.

Approach 2: The delay assigned to each flight is the average of the total system delay. System delay is obtained by cumulating individual flight delays. The primary difference in this reward structure lies in the case where the actual delay absorbed by the flight is larger than the required delay. In Approach 2, unlike Approach 1, the multi-agent group is still rewarded positively for the flight's successful absorption of delay. However, the multi-agent group is penalized by a factor of the amount of delay that was exceeded by the flight. The reason for not fully penalizing the group is to cater for flights that may not be able to fulfil their delay due to the 10% limit in speed modification.

Algorithm 1 Reward Approach 1

```
function ComputeReward($S$,$A$,$R$) ▷ Where S - Scheduled time, A - Actual time, R - Required time
    if $A > R$ then
        $reward \leftarrow -50$    ▷ Penalised constant value
    else if $S < A \leq R$ then
        $reward \leftarrow A - S$
    else
        $reward \leftarrow -(S - A)$    ▷ Penalised for not absorbing delay
    end if
end function
```

Algorithm 2 Reward Approach 2

```
function ComputeReward($S$,$A$,$R$) ▷ Where S - Scheduled time, A - Actual time, R - Required time
    if $A > R$ then
        $reward \leftarrow A - S$
        $reward \leftarrow reward - (A - R)/10$    ▷ Penalised for exceeding maximum delay
    else if $S < A \leq R$ then
        $reward \leftarrow A - S$
    else
        $reward \leftarrow -(S - A)$    ▷ Penalised for not absorbing delay
    end if
end function
```

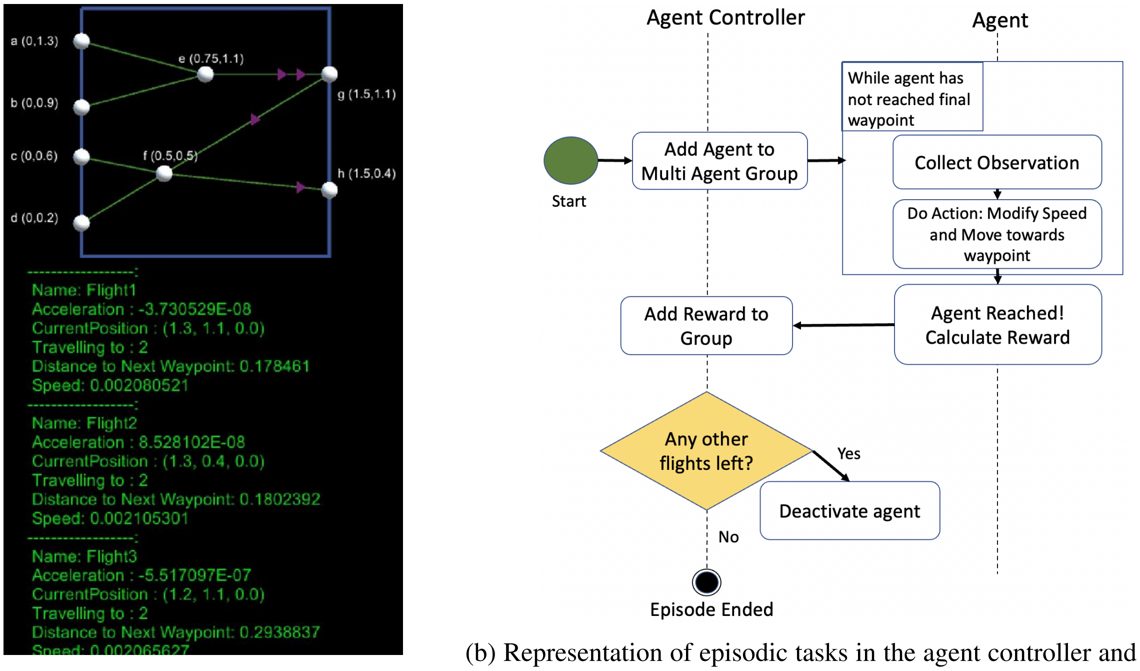
4 SIMULATION

4.1 Episodic Training of MAS

The tasks taking place in the Simulator were Episodic in nature, indicating that the tasks had a final or a terminal state. This implies that once a flight finished its path and had reached the end goal (last waypoint), the flight waited for every other flight in the flight plan to finish their paths before the episode was re-started. Thus, the episode's commencement was not on an individual agent basis, but rather dependent on the entire multi-agent group. In order to facilitate the requirements of the MAS, as discussed in Section 3, two classes were utilised to control the behaviour of the MAS - *Agent* and *AgentController*. The episodic interaction of the two classes can be highlighted in Figure 5b. The Multi-Agent Posthumous Credit Assingment (MA-POCA) trainer was introduced by Unity Technologies (Cohen et al. 2021). MA-POCA utilises the Independent Actor with Centralized Critic (IACC) framework, where a critic that is trained on joint information is utilized for updating independent agents or actors. The MA-POCA approach was tested with other trainers such as Proximal Policy Optimzation (PPO), and COunterfactual Multi-Agent Policy Gradients, and consistently produced a larger episodic reward for the same number of steps. Thus, this simulation uses the MA-POCA trainer provided by the Unity ML Agents Toolkit. Figure 5a shows the view of the simulator while the training was taking place, with the flight details displayed below the visual movement of the flights in the airspace boundary.

4.2 Hyper-parameter selection

Prior to training the MAS Simulation, hyperparameters defaulted to the standard were changed and compared to determine the best set of parameters that were both, computationally cost-effective, and yielded good results. By obtaining a balance between accuracy and computational cost, these hyper-parameters were finalized to eventually train and compare the two reward approaches. Comparisons were first done to



```
------------------:
Name: Flight1
Acceleration : -3.730529E-08
CurrentPosition : (1.3, 1.1, 0.0)
Travelling to : 2
Distance to Next Waypoint: 0.178461
Speed: 0.002080521
------------------:
Name: Flight2
Acceleration : 8.528102E-08
CurrentPosition : (1.3, 0.4, 0.0)
Travelling to : 2
Distance to Next Waypoint: 0.1802392
Speed: 0.002105301
------------------:
Name: Flight3
Acceleration : -5.517097E-07
CurrentPosition : (1.2, 1.1, 0.0)
Travelling to : 2
Distance to Next Waypoint: 0.2938837
Speed: 0.002065627
```

(a) Visualisation of MAS training on Unity.

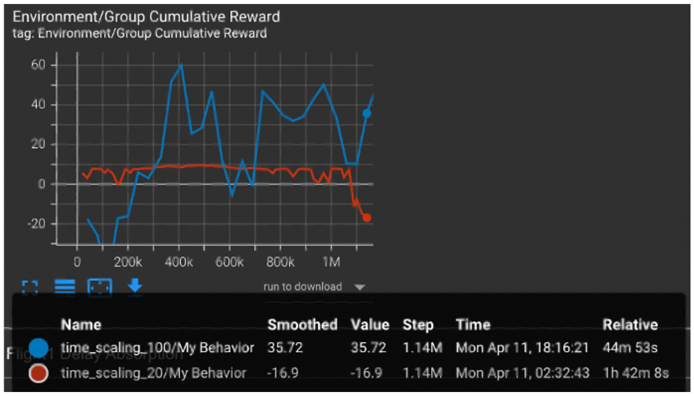
(b) Representation of episodic tasks in the agent controller and agent classes.

Figure 5: Episodic Simulation of the MAS. Panel (a) shows a sample simulation taking place. Panel (b) depicts the framework governing the interaction between the Agent Controller class, which dealt with the Multi-Agent group as a whole, and the Agent class, which represented each individual agent.

determine the most optimal buffer size, batch size, and time scale. Buffer Size and Batch Size are important numeric hyperparameters that influence the training processes for the MAS Training. Another parameter that was considered was the timescale, which is defaulted to 20 to speed up training. It was noted that changing the time scale did not produce any irregularities with the accuracy of the simulations, in terms of the ETAs and time calculations done for the flights. In order to obtain the best parameters, two experiments were conducted: (i) Batch Size – 4096, Buffer Size – 40960, Time-Scale – 20 (ii) Batch Size – 1024, Buffer Size – 10240, Time-Scale – 100.

Figure 6 highlights that increasing the time scale drastically decreased the training time, from a relative 1hr 42 min for 1.14M steps with time scaling set to 20, to 44 min 53 seconds for 1.14M steps. Moreover, Figure 6 shows that while the updates with the buffer and batch sizes of 40960 and 4096 (Red line) were more stable, the configuration with the lower buffer and batch sizes of 10240 and 1024 respectively, learned the policy a lot quicker, as it yielded a higher cumulative reward. Thus, the chosen values for buffer, batch sizes, and time scale corresponded to the values in Comparison 1 (Blue).

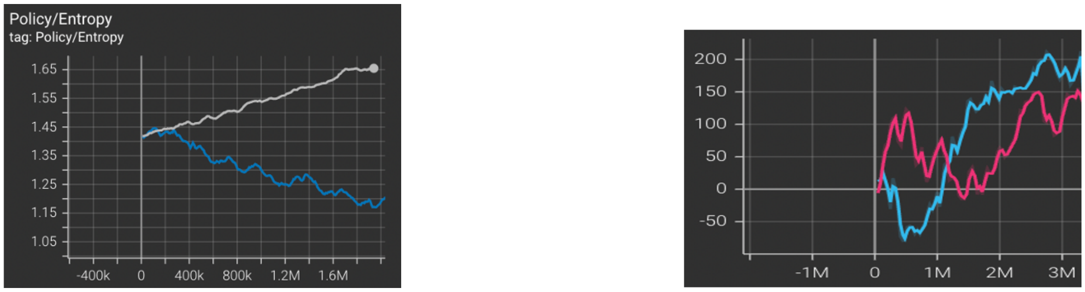
The beta parameter, which directly affects the entropy that exists with respect to the standard policy function created by the agent's actions, was also experimented on. Trainings were conducted using beta values of 1e-3 and 1e-2, and the changes in the policy's entropy were compared. In a successful training, the entropy of the policy should decrease as time goes on. Figure 7a shows that the Grey Graph (Comparison 2), with a higher beta value of 1e-2, showed an undesired increase in entropy of the policy, indicating that the model faced difficulties learning a trend in the policy, and the randomness in action choices continued to increase. The blue line graph, however, displays a steady decrease in entropy, as desired. Thus, the beta parameter of 0.001 (Blue), was chosen as the final beta hyperparameter. The next parameter that was compared and tested was the learning rate. The learning rate determines the strength of each update in the gradient descent. The hyperparameters compared to determine the optimum learning rate were 0.001 and 0.0005. As depicted by the Cumulative Reward comparison in Figure 7b, the blue line recognized the


Environment/Group Cumulative Reward
tag: Environment/Group Cumulative Reward
60
40
20
0
-20
0
200k
400k
600k
800k
1M
run to download

	Name	Smoothed	Value	Step	Time	Relative
●	time_scaling_100/My Behavior	35.72	35.72	1.14M	Mon Apr 11, 18:16:21	44m 53s
●	time_scaling_20/My Behavior	-16.9	-16.9	1.14M	Mon Apr 11, 02:32:43	1h 42m 8s

Figure 6: Comparing buffer sizes and time scales - large buffer (Blue) and low buffer (Red).

drop in negative rewards quicker than the pink line, and started producing increase in cumulative reward by 0.5 M steps. The cumulative reward was compared for the first 5 million steps, and it was determined that a learning rate of 0.001 eventually steadied the system to yield a higher reward. Thus, a learning rate of 0.001 was chosen for training the two reward approaches.



(a) Entropy comparison for ”Beta” - trained by lower learning rate (pink), and parameter - 1e-3 (Blue) and 1e-2 (Gray).

(b) Comparison of cumulative reward obtained by lower learning rate (Pink), and larger learning rate (Blue).

Figure 7: Hyperparameter comparisons for (a) entropy and (b) learning rate.

5 RESULTS

The results obtained by the two reward structures were compared in terms of the cumulative reward obtained by the training, and the percentage of delay absorption for all the flights.

5.1 Cumulative Reward Received by MAS

Figure 8 displays the Cumulative reward obtained by the environment for 5 Million steps of training. Approach 1 (Red) yielded a generally higher cumulative reward, as compared to Approach 2 (Orange). At approximately 3.6M steps, the system experiences a stark decrease in the Cumulative Rewards for Approach 1. This is in line with the reward system provided to the training. As the training improves, the model in Approach 1 aims to maximize the delay absorption by the flight. However, when the actual delay absorption reaches a value that is greater than the required delay absorption, the whole system is penalized – thus justifying the start decrease in rewards. The policy recognizes this and reduces the actual delay time to avoid the penalty in the next iterations. Approach 2, on the other hand, is not penalized by a large amount if the actual delay value is greater than the desired delay. In this case, Approach 1 did a better job in maximizing the delay absorption and maximizing system-level awards.

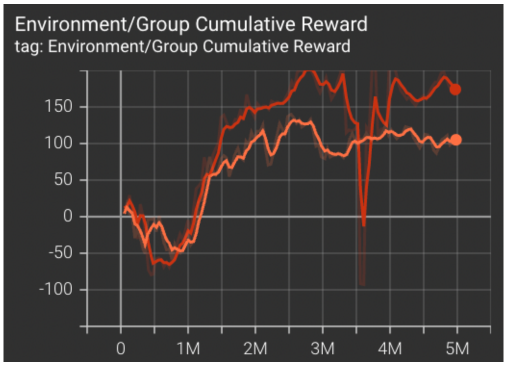


Figure 8: Cumulative rewards obtained by approach 1 (Red) against approach 2 (Orange).

5.2 Comparison of Delay Percentage Absorbed by Flights

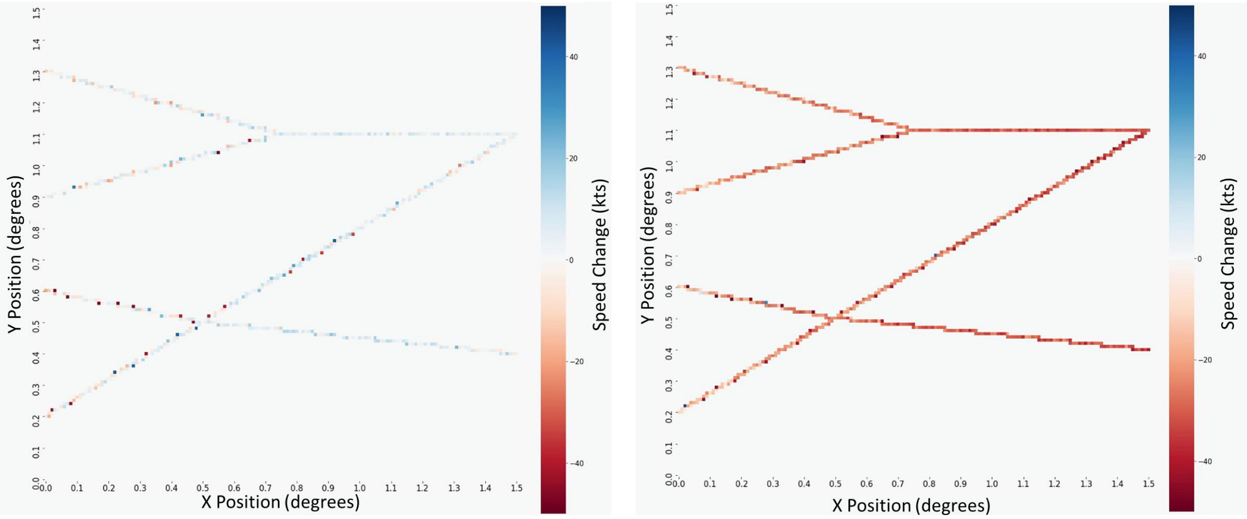
According to Figure 8 , the maximum Cumulative Reward was achieved at step 2.75M for Approach 1, and at 2.53M for Approach 2. The delay absorbed by approaches 2 and 1 for each flight have been formulated by Table 1 to determine the delay absorption that yielded the maximum cumulative reward. As seen from 1, Approach 1 consistently produced a greater percentage of delay absorption in the flights. The change in the reward could have been a key cause of slower learning in Approach 2. This approach requires a larger amount of steps to understand policy changes during training, which might be more apt for situations with greater complexity.

Table 1: Comparison of delay absorbed by the two approaches at maximum cumulative reward.

	Delay Absorbed by Approach 1 at Maximum Cumulative Reward (%) (2.75 Million Steps)	Delay Absorbed by Approach 2 at Maximum Cumulative Reward (%) (2.53 Million Steps)
Flight 1	94.76	69.33
Flight 2	79.28	40.27
Flight 3	90.39	65.19
Flight 4	65.38	40.42

5.3 Flight Plan Creation

The results for each trained episode were saved in an excel file, in terms of the original speed of the flight, the modified speed of the flight, and the change in flight speed. These results were obtained at Episode 0 and Episode 70 for Approach 1, which produced better results in terms of the total, system-level delay absorbed. In the heatmaps displayed in Figure 9a and Figure 9b, an absorption of delay/decrease in the flight's original speed is represented by the blue color of the cell. As it can be seen in the heatmaps below, the decrease in speed and increase in delay absorption can be seen by the color change from Episode 0 to Episode 70. The moderated speed at most states of the agent was lower than the original speed of the flight in Episode 70 (as indicated by the red color of the graph), indicating speed reduction in the system. This validates the policy function that was trained by the model, and confirms that the approach succeeds in absorbing delays while avoiding any potential collisions in the flight plan. It is also possible to visualize the pattern in which flight delays are absorbed. For instance, a large amount of delay was absorbed between the waypoints on the upper right link of the flight plan, since greater speed reductions took place during that phase as indicated by the dark red colour of the heatmap.



(a) Speed moderation heatmap - Episode 0.

(b) Speed moderation heatmap - Episode 70.

Figure 9: Speed moderation heatmaps at episode 0 and episode 70 of MA-POCA Training. At episode 0, the agents had just started learning and the heatmap resulted in largely light blue cells, indicating that speed reductions were not taking place. By episode 70 of the training, the reward system helped agents learn the need for speed reduction, resulting in a red heatmap that indicated speed reductions in the estimated range of 0 - 40 kts. This shows a successful MA-POCA training, as speed reductions aid in delay absorption.

6 CONCLUSION AND FUTURE WORK

In conclusion, the development of a multi-agent system and the implementation of RL techniques largely allowed flights to absorb delays in the range of 81-95% amongst all flights using Approach 1. Approach 2 provided poorer results as compared to Approach 1, significantly due to Approach 2's direct focus on creating a balance between two main objectives - speed reductions, and distributed delay absorption by flights. The involvement of two objectives in Approach 2, as compared to one main objective in Approach 1, could have compromised on the flights' amount of delay absorption. While Approach 2 provided lower speed absorption for the same amount of training steps, it could be further utilized and explored for more complex scenarios where flight delay distribution is more uneven. The speed heatmaps produced in Figure 9 validated the reward structure that had been adopted for the simulation, as a positive reward proportional to the delay absorbed by the flights encouraged flights to absorb a larger amount of delay. The implementation of RL may be a more apt means of airspace simulation, largely due to the ability of RL to adapt to dynamic changes in the agent's environment. Moreover, the system ensured that a cooperative reward system was built, such that all flights aimed towards the common goal of achieving the maximum possible system delays. This was further highlighted in the stark drop of environmental reward in Approach 1, when the delay absorbed by a flight was larger than the required delay, implying that if a flight unfairly absorbs more delay than designated, the whole multi-agent group is penalised. While flights experienced moderations in speeds in the 10% range, it was ensured that collisions in the airspace do not take place due to such an issue. This method provided consistently positive results in terms of fulfilling the purpose of the MAS to potentially maximize delay absorption amongst flights.

Future work would entail tuning the model in terms of its hyper-parameters and reward systems. In this case, solely extrinsic rewards in Unity 3D were considered to train the model. However, other rewards provided, such as curiosity-based reward signals, Random Network Distillation (RND), or imitation learning techniques, could also be combined with the current model to improve results. Moreover, more complex and realistic scenarios could also be simulated. In cases where the 10% speed modification limit prevents it from achieving its maximum required delay, implementations for the flight to automatically communicate with another flight could be addressed in the future to meet the required system-level delay absorption.

ACKNOWLEDGEMENTS

This project is supported by the National Research Foundation, Singapore, and the Civil Aviation Authority of Singapore, under the Aviation Transformation Programme. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and the Civil Aviation Authority of Singapore.

REFERENCES

Airports Council International 2021. "The Impact of COVID-19 on the Airport Business and the Path to Recovery". https://aci.aero/2021/03/25/the-impact-of-covid-19-on-the-airport-business-and-the-path-to-recovery, accessed 12th January 2022.

Brilliant.org 2022. "Markov Chains". https://brilliant.org/wiki/markov-chains, accessed 28th January.

Carlier, S. et al. 2007. "Environmental Impact of Air Traffic Flow Management Delays". In *Proceedings of the USA/FAA Air Traffic Management Research and Development Seminar 2007*. July 2nd-5th, Barcelona, Spain, 1-13.

Cohen, A. et al. 2021. "On the Use and Misuse of Absorbing States in Multi-agent Reinforcement Learning". *arXiv preprint arXiv:2111.05992*.

Delgado, L., and X. Prats. 2012. "En Route Speed Reduction Concept for Absorbing Air Traffic Flow Management Delays". *Journal of Aircraft* 49(1):214–224.

Dhief, I. et al. 2020. "Speed Control Strategies for E-AMAN Using Holding Detection-Delay Prediction Model". In *Proceedings of the 10th EUROCONTROL SESAR Innovation Days 2020*. December 7th-10th, Virtual Conference, 1-10.

International Civil Aviation Organization 2019. "The World of Air Transport in 2019". https://www.icao.int/annual-report-2019/Pages/the-world-of-air-transport-in-2019.aspx, accessed 4th March 2022.

Ma, J. et al. 2019. "Integrated Optimization of Terminal Maneuvering Area and Airport at the Macroscopic Level". *Transportation Research Part C: Emerging Technologies* 98:338–357.

Matsuno, Y. et al. 2020. "Analysis of achievable airborne delay and compliance rate by speed control: A case study of international arrivals at Tokyo international airport". *IEEE Access* 8:90686–90697.

SESAR 2015. "Extended Arrival Manager (E-AMAN) - SESAR Joint Undertaking". https://www.sesarju.eu/sites/default/files/documents/wac2015/E-aman_factsheet_FINAL.pdf, accessed 10th January 2022.

SESAR 2022. "Extended AMAN". https://skybrary.aero/articles/extended-aman, accessed 10th January 2022.

Steimle, L. N. et al. 2021. "Multi-model Markov Decision Processes". *Institute of Industrial and Systems Engineers Transactions* 53(10):1124–1139.

AUTHOR BIOGRAPHY

KANUPRIYA MALHOTRA completed her B.Eng degree in Mechanical Engineering with a minor in Computing and Data Analytics. Her research experience involves working on optimizing routing and scheduling algorithms for postal delivery via Unmanned Aerial Vehicles in Singapore. She has contributed to researches done on the implementation of computer vision and artificial intelligence in applications such as automated waste sorting and autonomous underwater vehicles. She is currently working at Micron as a Data Analytic Engineer. Her email address is kanupriyamalhotra1212@gmail.com.

ZHI JUN LIM received the B.Eng degree in Aerospace Engineering and B.A degree in Economics from Nanyang Technological University, Singapore, in 2019. She is currently pursuing the Ph.D. degree with the Air Traffic Management Research Institute, Nanyang Technological University, Singapore. Her research work focuses on machine learning and artificial intelligence with applications in air traffic management. Her email address is zhijun001@e.ntu.edu.sg.

SAMEER ALAM received PhD in Computer Science with specialization in Machine Learning from the University of New South Wales, Canberra, Australia, in 2008. He is currently the Deputy Director of Air Traffic Management Research Institute and Associate Professor, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore. His email address is sameeralam@ntu.edu.sg.