Fast Decision Support for Air Traffic Management at Urban Air Mobility Vertiports using Graph Learning

Prajit KrisshnaKumar[1*], Jhoel Witter[2*], Steve Paul[3*], Hanvit Cho[4*], Karthik Dantu[5‡], and Souma Chowdhury[6*†]

***Abstract*— Urban Air Mobility (UAM) promises a new dimension to decongested, safe, and fast travel in urban and suburban hubs. These UAM aircraft are conceived to operate from small airports called vertiports each comprising multiple take-off/landing and battery-recharging spots. Since they might be situated in dense urban areas and need to handle many aircraft landings and take-offs each hour, managing this schedule in real-time becomes challenging for a traditional air-traffic controller but instead calls for an automated solution. This paper provides a novel approach to this problem of Urban Air Mobility - Vertiport Schedule Management (UAM-VSM), which leverages graph reinforcement learning to generate decision-support policies. Here the designated physical spots within the vertiport's airspace and the vehicles being managed are represented as two separate graphs, with feature extraction performed through a graph convolutional network (GCN). Extracted features are passed onto perceptron layers to decide actions such as continue to hover or cruise, continue idling or take-off, or land on an allocated vertiport spot. Performance is measured based on delays, safety (no. of collisions) and battery consumption. Through realistic simulations in AirSim applied to scaled down multi-rotor vehicles, our results demonstrate the suitability of using graph reinforcement learning to solve the UAM-VSM problem and its superiority to basic reinforcement learning (with graph embeddings) or random choice baselines.**

I. INTRODUCTION

Urban Air Mobility (UAM) based on electric Vertical Takeoff and Landing (eVTOL) aircraft is becoming an increasingly popular concept for future urban transportation, with the objective to mitigate problems such as traffic congestion and circuitous routes. Such eVTOLs are electric aircraft with the ability to transport 2-5 passengers, air-ambulance or emergency services, or serve in a goods transport role. Over the past few years, there have been numerous feasibility studies on UAM in the context of addressing traffic congestion, and its overall impact on reducing carbon emissions [1], [2], [3]. However, the realization of UAM comes with its own set of challenges which includes the design challenges with regards to vertiport infrastructure, accessibility and noise impact, and operational challenges which include UAM fleet scheduling [4] and vertiport operation [5]. In these contexts, safety is the primary driving factor behind the planning processes [5]. A Vertiport or aerodrome is conceived as a site where the eVTOLs take off, land, and charge their batteries [6]. In this work, we focus on the problem of vertiport operation, specifically real-time scheduling of eVTOL take-off and landing. The key objectives are to ensure safety (i.e., avoid collisions), minimize delays with respect to a pre-defined schedule of takeoff and landing, and conserve battery. This problem is named UAM-Vertiport Schedule Management (UAM-VSM).

Once deployed, such UAM vehicles are expected to be entering or leaving a vertiport located in a dense urban environment [7], somewhat similar to a subway network in major cities, and unlike typical airports used in general aviation. Unlike in ground transportation, minor collisions or undesirable proximity can lead to dire consequences, thereby necessitating strict spatiotemporal constraints on their operation. Assuming high traffic at any given vertiport (which is expected for an economically viable UAM network) and uncertainties due to weather and system failures, over-reliance on a human Air Traffic Controller (ATC) to perform the real-time vertiport traffic management is unduly risky. The high frequency of decision-making could lead to a cognitive overload for the ATC and result in poor decision-making [8]. Hence, there is an urgent need to develop *automated* approaches to manage the real-time scheduling of takeoff/landing of eVTOL aircraft at vertiports.

Prior works on UAM scheduling have mostly focused on eVTOL fleet scheduling and path planning across multiple vertiports in a region [9], [10] to maximize profit and meet potential travel demand. Very limited quantitative work exists in automated decision-support for real-time scheduling of takeoff/landing with the objectives to maximize safety and minimize delays. Unlike the other works which schedule higher-level decisions such as which destination/vertiport to visit next, or when to charge an eVTOL [11], here we assume that a high-level schedule over a longer time horizon of say 6-12 hours already exists. Our focus is on computing real-time decisions during each 1-minute time window, during which tasks (land, takeoff, move to spot, etc.) have to be allocated to up to 4 aircraft within the operational air space managed by that vertiport. In addition, unlike prior works where the eVTOL is simulated by a simple linear model [12], we consider a more realistic eVTOL simulation developed in AirSim [13], based on a scaled-down multi-rotor aerial

† Corresponding Author, soumacho@buffalo.edu

*Department of Mechanical and Aerospace Engineering, University at Buffalo, Buffalo, NY {prajitkr, jhoelwit,stevepau,hanvitch}@buffalo.edu

‡Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY kdantu@buffalo.edu

This work was supported by Stephen Still Institute for Sustainable Transportation and Logistics (SSISTL) and National Science Foundation (NSF) award CMMI 2048020. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the SSISTL and NSF.

vehicle model. This is done to ensure that the simulated outcomes have a reduced reality gap. Moreover, this work is among the first to also present a scaled-down physical validation of the proposed solution to UAM-VSM, through indoor experiments with four small quadcopters.

Due to high computational costs, traditional methods such as Mixed Integer Non-Linear Programming (MINLP) and heuristic-search-based methods are unviable for the real-time decision-making needs of the UAM-VSM problem. Hence a learning-based method is posited as a robust alternative, which can yield policies that are real-time executable. Specifically, we take a novel graph-learning approach to solving this problem in a scalable and generalizable manner. To do so, firstly we consider the different designated physical spots (Figure 2) that can be allocated to a vehicle, such as landing/take-off pad (or spot), hovering spot and charging spot or battery port, as nodes in a graph. We also consider the eVTOLs aircraft to be represented as another *Graph*. With this representation, we exploit Graph Neural Network (GNN) for feature extraction over the graph-encoded state-space, and Proximal Policy Optimization (PPO) [14] is used to train this policy network.

Key Contributions: The primary contributions of this work lie in: 1) Formulating the UAM-VSM problem as a Markov Decision Process (MDP) and development of a relatively realistic virtual environment to simulate this MDP for scaled-down multi-rotor vehicles; 2) Testing the hypothesis that a graph abstraction of the state-space of designated locations in the vertiport's airspace and vehicles being managed, along with graph learning, can yield effective policies for this problem; 3) Analyzing the trade-offs between objectives such as takeoff delay, landing delay, collision avoidance, and battery conservation, achieved with such graph-learning based UAM-VSM policies. Contribution 2 is demonstrated via comparisons against vanilla RL solutions, randomized decision-making, and first-come-first-serve decision-making, with all methods evaluated over a sampling of different 24-hour operational periods.

The rest of this paper is structured as follows. In Sec. II, some of the prior works related to the problem is presented. Sec. III describes the formulation of UAM-VSM as an MDP, followed by the description of the proposed state abstraction and graph learning architecture, in Sec. IV. In Sec. V, we present the simulation environment developed for Vertiport management problems. Sec. VI presents the numerical experiments and comparison of results with that of baselines, namely a random method, a first-come-first-served method, and a standard RL approach. Sec. VII summarizes our concluding remarks.

II. Related work

In commercial aviation, the most common approach for aircraft takeoff/landing schedule is the First-Come-First-Served (FCFS) approach [15]. The main advantage of this approach is that it is easy to implement without the need for any sophisticated scheduling software. The only constraint for this approach is maintaining a safe distance. However,

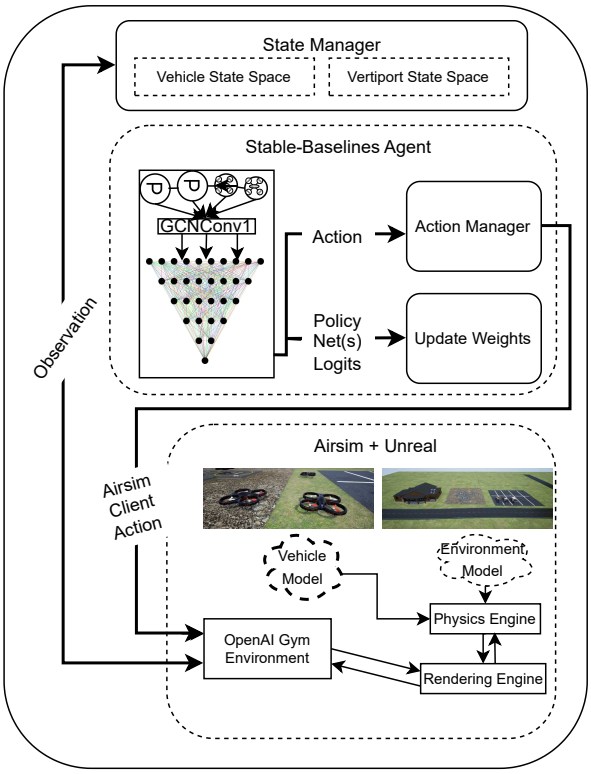

Fig. 1. Overview of Simulation Environment displaying the progression of Information through the State Manager, Action Manager, Learning Policy Network, and the Simulation Engine (Airsim and Unreal Engine)

the major drawback of this approach is that it does not provide any scope for improvement to decrease time delays. Trivizas and Lieder et al. [16], [17] proposed a Dynamic Programming algorithm for optimal landing on a runway, while Beaslet et al. and Abela et al. [18], [19] proposed a Mixed Integer Programming (MIP). These approaches are feasible only for commercial aviation takeoff/landing where the number of schedules is of the order of 1 every few minutes. In recent years, RL algorithms with Graph Neural Networks or GNN are being increasingly used to solve planning and scheduling problems such as TSP, VRP, Max-Cut, Min-Vertex, and MRTA [20], [21], [4], [22], [23], [24], [21], [10], [25]. Some of the recent works [10], [26], [9], [27] on UAM mainly focus on scheduling routes between vertiports with the aim to optimize an entity such as profit, delay, and demand met. Here, we build on our prior work on this under-addressed problem [28], by considering the critical objective of safety and presenting comparisons with a standard RL implementation that does not use graph abstractions.

III. Problem Formulation

We consider this an Urban Air Mobility-Vertiport Schedule Management (UAM-VSM) problem, where the goal is to design an automated centralized Air Traffic Controller (ATC) agent in order to successfully: **i.)** assign tasks to incoming vehicles (taking off to destinations, landing, hovering); **ii.)** maintain a sufficient battery level among all vehicles being controlled; **iii.)** ensure a safe flight for each vehicle without

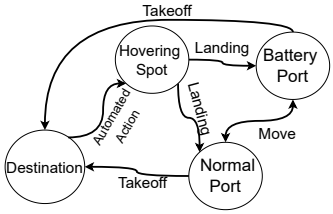

Fig. 2. State-Transition Diagram

collisions. Throughout this paper, we assume there is no communication loss and no uncertainties in the environment. To train the agent, we created an environment identical to that of a realistic vertiport, with two ports for landing (Normal Ports), and one charging station (Battery Port). There are four hovering spots arranged around the ports, as well as five destinations away from the agent's airspace that the vehicles travel to and from. The overall framework is shown in Figure 1. Each vehicle has 4 states and the state-transition diagram is shown in figure 2.

The environment is initiated with 4 vehicles at a random state and with a random schedule. Takeoff time is issued between 10 and 20 minutes after the vehicle returns to the vertiport, and likewise, the landing time is issued between 5 and 15 minutes after the vehicle begins traveling back to the vertiport from a destination. Each vehicle starts with a full battery and discharges at a rate of *discharge_rate* (Ω) per step. Ω is calculated based on the action:

$$\Omega = \begin{cases} d_t & \text{if cruising} \\ 0.5 & \text{if hovering} \\ 0.25 & \text{if idling on ground} \end{cases} \quad (1)$$

where d_t is the distance from one point to the next. When a vehicle rests at a charging station, it regains 10% battery per time-step. Thus a vehicle can lose battery even while not moving, which will encourage the agent to travel and engage with the charging port.

Vehicles move independently, allowing multiple simultaneous movements. This setup helps the agent learn collision avoidance when vehicles intersect. At each time step, the agent selects an available vehicle at the port to act. If the chosen vehicle has reached its destination, the agent waits until it re-enters the vertiport airspace before acting. The simulation runs 300 times faster than in real-time, so every second in the simulator corresponds to 5 minutes in real-time and is useful for training, as it allows for step times as low as 0.2 seconds, or 288 seconds (4.8 minutes) per episode (1440 steps). Every episode simulates a full day of operation. Every vehicle is updated with a minimum frequency of $45Hz$, which includes updating all features (location, delay, battery status, and vehicle status).

A. MDP Formulation

The Urban Air Mobility-Vertiport Schedule Management problem is modeled as a Markov Decision Process (MDP), where the state space consists of the state of the vertiports and the state of the vehicles. The vertiport state information is represented as graph $\mathscr{G}_{VP}$=(V_{VP}, E_{VP}), where V_{VP} represents the vertices or nodes of the graph and E_{VP} represents the set of edges. Every node $i \in V_{VP}$ has the following properties (also described in table I) δ_i^{VP} = $[P_a^i,\ P_t^i,\ x_i,\ y_i]$. Similarly, the vehicle state information is represented as a graph $\mathscr{G}_{EV}$=(V_{EV}, E_{EV}), where V_{EV} represents the nodes of the graph and E_{EV} represents the set of edges. Every node $i \in V_{EV}$ has the following properties (also described in table I) δ_i^{EV} = $[c_i,\ b_i,\ l_i,\ x_i, y_i]$. While there are no explicit environment uncertainties here, state transition in principle is not deterministic due to the possibility of aircraft collision (that triggers avoidance actions) modeled by the simulation.

TABLE I

MDP FORMULATION

Type	Variable
Vertiport States	Availability - P_a Port type - P_t Location - (x_p, y_p)
VTOL States	Current status - c_i Battery capacity - b_i Schedule status - l_i Location - (x_i, y_i)
Action Space	Stay still Takeoff Move/ land in normal port - 1,2 Move/ land in battery port - 1 Move to hover spots - 1,2,3,4 Continue previous action Avoid collision

The state and action space can be found in table I, and the reward is shown in equation 2. Here τ is the takeoff coefficient, γ is the landing coefficient, λ is the battery coefficient, β is the delay coefficient, § is the safety coefficient, and w_n are the weights.

$$R = w_1\tau + w_2\gamma + w_3\lambda + w_4\beta + w_5\S \quad (2)$$

Here are more details about the reward terms:

1) Takeoff-Landing coefficient: We classify a "**good**" takeoff as one where the vehicle is: **i)** departing punctually (within 5 minutes of its planned takeoff time); **ii)** taking off with a battery level exceeding 30%. The same standards apply to a "**good**" landing, with the added option that the vehicle can decide to land before its scheduled time. Both τ and γ fall within the range of $-5, 5$.

2) Battery coefficient: This coefficient is defined as:

$$\lambda = \begin{cases} 5 \times \frac{b_i}{100} & \text{if } b_i \geq 30 \\ -5 & else \end{cases} \quad (3)$$

The value gradually rises as the vehicle charges. To discourage operations at critical battery levels, a penalty is imposed once the battery capacity falls below 30%. In order to maintain the battery coefficient, the agent must ensure each vehicle is fully charged.

3) Delay coefficient: The delay time is computed from the point when the vehicle misses its window for either taking off or landing. This delay time accumulates till the vehicle receives a new schedule. The delay coefficient is calculated by:

$$\beta = -5 + 10 \times e^{-\Delta_i} \quad (4)$$

where Δ represents the delay in minutes. This formulation encourages the agent to keep the delay as low as possible by maximizing the delay coefficient.

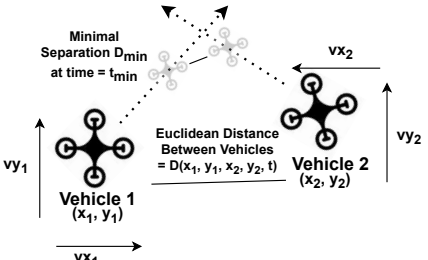

Fig. 3. Minimal Separation Scenario

4) Safety coefficient: Before the safety coefficient can be calculated, the environment will check to see if the selected vehicle: **i.)** is currently en-route to a location; **ii.)** has an intersecting path with another vehicle that is en route. This can be visualized in figure 3.

In the figure, two vehicles are moving toward a common intersection point. The distance between them at any given point is made into a function of time using the Euclidean distance combined with their immediate position and velocity vectors:

$$D(t) = \sqrt{(x_1 - x_2 + v_{x_1}t - v_{x_2}t)^2 + (y_1 - y_2 + v_{y_1}t - v_{y_2}t)^2} \quad (5)$$

where $x_n, y_n, v_{x_n}, v_{y_n}$ are the position and velocity components of vehicles 1 and 2. This equation is then differentiated with respect to time and solved for the local minimum, t_{min}:

$$t_{min} = -\frac{2(v_{x_1} - v_{x_2})(x_1 - x_2) + 2(v_{y_1} - v_{y_2})(y_1 - y_2)}{2(v_{x_1} - v_{x_2})^2 + 2(v_{y_1} - v_{y_2})^2} \quad (6)$$

t_{min} is then plugged back into equation 5 to get the minimum separation D_{min}. Each simulated vehicle has an occupant space of 1x1 meters, so if D_{min} is less than 3 meters and the agent doesn't take evasive action(avoid collision), the agent will be penalized:

$$\S = \begin{cases} 0 & \text{if vehicle is on the ground} \\ -5 & \text{else if } D_{min} \leq 3.0 \ \& \ action \neq avoid\ collision \\ 5 & \text{else if } D_{min} \leq 3.0 \ \& \ action = avoid\ collision \end{cases} \quad (7)$$

5) Reward weights: Each coefficient is multiplied by a weight $w_1, w_2, ..w_n$, reflecting the significance of each coefficient. In our scenario, safety is considered the most important, hence the highest weight is allocated to the safety coefficient §, followed by $\beta, \tau, \gamma, \lambda$, in that order.

IV. Learning Architecture

This paper focuses on a deep reinforcement learning framework-PPO [14]- utilizing a GCN for feature abstraction and a Multi-Layered Perceptron (MLP) for prediction.

A. Network Parameters

Linear layers with biases are used for all three networks (feature abstraction, value, and policy) except for the feature abstraction network in the GRL agent, for which GCNs are used. We make use of LeakyReLU with a slope of 0.1 (for feature abstraction and value network) & Tanh (for the policy network) activation layers, as they help reduce sparse gradients. Adam optimizer with a learning rate of 1e-5 is used for back-propagation. We chose Proximal Policy Optimization (PPO) for the reinforcement learning algorithm [14]. PPO is based on Trust Region Policy Optimization (TRPO) [29], and has an objective function tailored to clip policy expansion and allows for a safer policy update.

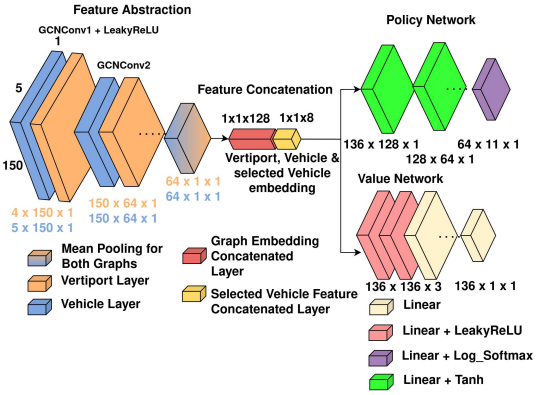

Fig. 4. The policy network along with the value network for the Graph Learning Architecture

B. Frameworks

Two frameworks are used for learning the policy model, namely the Multi-layer perceptron based RL network and the graph learning-based RL network Both frameworks have feature abstraction, policy, and value networks to work with PPO. The main difference lies in the feature abstraction network where: the RL agent uses a condensed feature vector with both the vehicle and vertiport state space; the GRL agent uses two GCNs, which take the vehicle and vertiport feature matrix along with their respective edge connectivity matrix. The GRL learning framework is shown in figure 4. Both policy networks will use a four-layer MLP with a log-softmax transformation to obtain log probabilities for the 11 actions. Both agents utilize masking which will depend on the state of the selected vehicle and the availability of each port. This takes away a layer of complexity and allows the agent to focus on other environmental factors, such as avoiding collisions and reducing delay.

V. Vertiport Simulation

The evolution of the gaming industry over the last decade has had a great impact on the robotics community, allowing us to simulate various complex tasks [30], [31], in a more realistic environment that is nearly impossible in real life due to the hardware limitations. To simulate the Vertiport, a custom 3D environment is developed using Unreal Engine [32] with AirSim [13] plugin being the backend for physics. AirSim is an open-source simulator plugin developed by Microsoft for unmanned aerial (UAV) and ground vehicles (UGV). Since the operation of VTOLs is similar to that of UAVs, throughout this paper we consider UAVs to be vehicles. Fig. 1 shows the overview of the developed simulation environment. Unreal Engine 4 is used as the GUI and AirSim plugin provides dynamics for the vehicle. To reduce the memory and computational footprint of the unreal engine, the entire environment has been created with 101k static triangles, limiting the memory to 3.4 Megabytes(excluding VTOLs). The OpenAI Gym interface provides a connection between Airsim, State/Action Managers, and the RL Agent. This interface receives raw data of all the vehicles from AirSim and filters the necessary data, which then will be transferred to the State Manager. The State Manager and the

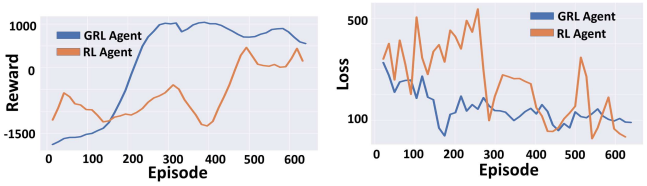

Fig. 5. Reward (left) and Loss (right) plots during training of the RL and GRL Agent for 632 Episodes

Action Manager inherit the properties of all the vehicles and ports. Filtering data and structuring the data takes place in the state manager, while the action manager decodes the data from the Policy model to AirSim understandable format.

VI. Results and Discussion

This section presents the results of the case studies performed for the UAM-VSM problem. The following subsections will delve into the **i.)** training analysis of the GRL and RL agent. **ii.)** a comparison of test results with: the GRL agent; the RL agent; a random agent; a First-Come-First-Serve (FCFS) agent. **iii.)** analyses of effects of safety parameter and weights in the reward formulation. Both agents received the same information in their observation space and used the same environment for a fair comparison. Here, we consider only 4 vehicles since the use of realistic simulations limits the number of vehicles that can be simultaneously modeled without blowing up the training process. Further, departing vehicles are not simulated anymore once they leave the port (within their allocated 20-min window), and instead their embodiment is taken over by another aircraft that is either landing or waiting for a take-off schedule, keeping the number of aircraft simulated at 4 at all times.

A. Learning Curve

The reward and loss for each agent during training are shown in figure 5. The GRL agent was able to converge, while the RL agent struggled. The GRL agent shows a stable loss decline, while the RL agent's loss exhibits more oscillations, indicating that the GRL agent found it easier to correlate its observations in the state space.

B. Baseline Comparison

In order to evaluate the performance of the GRL agent, we compare it with an RL agent with a Multi-Layer Perceptron (MLP) as feature extractors (instead of GNN) trained with the same parameters referenced in section IV, a random agent, and a First-Come-First-Served (FCFS) agent (adapted from [33]). The random agent chose random feasible actions from the action space during each decision-making instance. In the FCFS approach, each vehicle ready for landing is put in a queue for recharging (for up to 6 steps), and the vehicle which is ready to take off after recharging is put into another queue. A scheduler commands each vehicle in the queues to execute its action one after the other. Figure 6 show the results after testing each agent for 50 episodes (72000 steps). It is important to note the delay depicted is cumulative across all 4 vehicles per episode. The GRL and FCFS agent are observed to have the best performances of the four when it comes to cumulative reward (GRL: 565 ± 522.7, FCFS: 293 ± 86), delay, good takeoffs, and battery management. This is expected, as the GRL agent learned the feature space faster and its goal is to maximize all metrics, while the FCFS agent takes off and lands at set intervals that align closely with each vehicle's schedule and maximize specific metrics (takeoffs, landings, and battery management). As for collisions, the GRL agent has a mean of 0.3 ± 1.7, which is 76% smaller than that of the RL agent and 90% smaller than the FCFS agent, which has means of 0.86 ± 4.24 and 2.88 ± 0.66 respectively. The RL agent does outperform the GRL agent in time scheduling efficiency, with a mean delay of 7.49 ± 1.43 hours per vehicle, which is an 80% lower standard deviation compared to the GRL agent. The random agent struggled to learn delay, collision, and battery metrics.

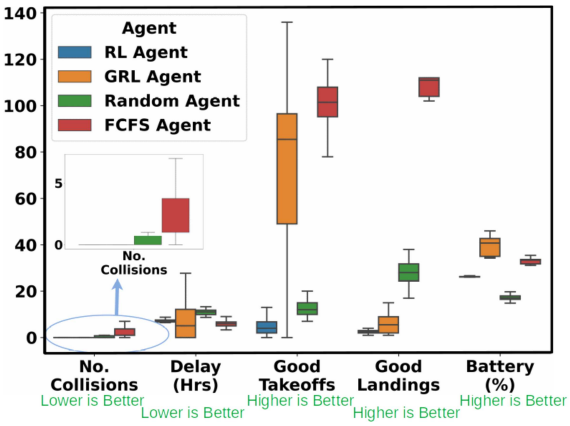

Fig. 6. Comparison of all the reward terms against the baseline methods.

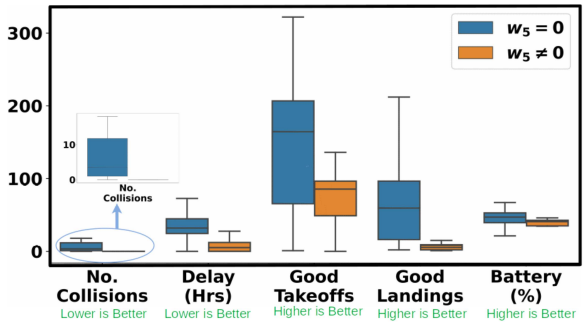

Fig. 7. Analysis of the effect of the Safety Weight on GRL Agent

C. Analysis of the Safety Coefficient

In this section, we analyze the importance of the safety parameter §. Here, we run the training for 2 different values of w_5 in the reward function (eqn. 2), $w_5 = 0$ and $w_5 = 2.2$. These models are then evaluated for 50 episodes and the results are shown in Fig. 7. The model trained without a safety weight: **i.)** got into more collisions over the course of an episode; **ii.)** experienced more takeoffs and landings, which led to it outperforming the model with a safety weight and also experiencing more delay. This analysis shows that more training time and variety is required to understand an environment with both delay and safety factors, as the agent has seemingly less difficulty learning one or the other and is forced to compromise when learning both.

VII. CONCLUSION

In this paper, we proposed a Graph RL (GRL) method to generate a policy that serves as the real-time air traffic control (ATC) system at a vertiport. Specifically, this policy performs real-time scheduling of take-offs and landings of VTOL aircraft, considering safety, delay, and battery level. We formulated the UAM vertiport management problem as an MDP with the state of vertiport and states of aircraft expressed as two separate graphs. We use two GCNs to extract the vertiport and aircraft state information as learnable feature vectors. The training was performed using PPO on a reward function defined as a weighted combination of landing/takeoff delay, number of collisions, and battery consumption, across the vehicles being managed. GRL policy performance is compared with three baselines: MLP-based RL agent, Random agent, and First-Come-First-Served. Due to GCN's better feature extraction, GRL outperformed MLP-RL, learning faster and achieving higher episodic rewards. In test episodes, GRL surpassed MLP-RL and the random agent, demonstrating its generalizability. In the future, simulation-based nonlinear optimization could enhance RL solutions, approximating the optimality gap. Improved draft interactions with nearby aircraft could also boost the effectiveness of the proposed UAM vertiport management approach.

REFERENCES

[1] A. R. Kadhiresan and M. J. Duffy, "Conceptual design and mission analysis for evtol urban air mobility flight vehicle configurations," in *AIAA Aviation 2019 Forum*, 2019, p. 2873.

[2] R. Goyal, C. Reiche, C. Fernando, J. Serrao, S. Kimmel, A. Cohen, and S. Shaheen, "Urban air mobility (uam) market study," Tech. Rep., 2018.

[3] H. Chao, A. Maheshwari, D. DeLaurentis, and W. Crossley, "Weather impact assessment for urban aerial trips in metropolitan areas," in *AIAA AVIATION 2021 FORUM*, 2021, p. 3176.

[4] S. Paul, P. Ghassemi, and S. Chowdhury, "Learning scalable policies over graphs for multi-robot task allocation using capsule attention networks," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8815–8822.

[5] A. VERTIPORT, "High-density automated vertiport concept of operations."

[6] M. Daskilewicz, B. German, M. Warren, L. A. Garrow, S.-S. Boddupalli, and T. H. Douthat, "Progress in vertiport placement and estimating aircraft range requirements for evtol daily commuting," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 2884.

[7] N. M. Guerreiro, G. E. Hagen, J. M. Maddalon, and R. W. Butler, "Capacity and throughput of urban air mobility vertiports with a first-come, first-served vertiport scheduling algorithm," in *AIAA Aviation 2020 Forum*, 2020, p. 2903.

[8] A. Mathur, K. Panesar, J. Kim, E. M. Atkins, and N. Sarter, "Paths to autonomous vehicle operations for urban air mobility," in *AIAA Aviation 2019 Forum*, 2019, p. 3255.

[9] S. G. Manyam, D. W. Casbeer, S. Darbha, I. E. Weintraub, and K. Kalyanam, "Path planning and energy management of hybrid air vehicles for urban air mobility," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 176–10 183, 2022.

[10] S. Paul and S. Chowdhury, *A Graph-based Reinforcement Learning Framework for Urban Air Mobility Fleet Scheduling*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2022-3911

[11] R. W. Simpson, "Computerized schedule construction for a vtol airbus transportation system." *Journal of Aircraft*, vol. 5, no. 3, pp. 299–305, 1968.

[12] J. P. Theron, J. F. Horn, and D. A. Wachspress, "An integrated simulation tool for e-vtol aeromechanics and flight control analysis," in *Aeromechanics for Advanced Vertical Flight Technical Meeting 2020, Held at Transformative Vertical Flight 2020*. Vertical Flight Society, 2020, pp. 115–130.

[13] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[15] A. R. Odoni, J.-M. Rousseau, and N. H. Wilson, "Chapter 5 models in urban and air transportation," in *Operations Research and The Public Sector*, ser. Handbooks in Operations Research and Management Science. Elsevier, 1994, vol. 6, pp. 107–150.

[16] D. A. Trivizas, "Optimal scheduling with maximum position shift (mps) constraints: A runway scheduling application," *The Journal of Navigation*, vol. 51, no. 2, pp. 250–266, 1998.

[17] A. Lieder, D. Briskorn, and R. Stolletz, "A dynamic programming approach for the aircraft landing problem with aircraft classes," *European Journal of Operational Research*, vol. 243, no. 1, pp. 61–69, 2015.

[18] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson, "Scheduling aircraft landings–the static case," *Transportation Science*, vol. 34, no. 2, p. 180–197, may 2000.

[19] J. Abela, D. Abramson, M. Krishnamoorthy, A. de Silva, and G. B. Mills, "Computing optimal schedules for landing aircraft," 1993.

[20] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 6348–6358.

[21] R. A. Jacob, S. Paul, W. Li, S. Chowdhury, Y. R. Gel, and J. Zhang, "Reconfiguring unbalanced distribution networks using reinforcement learning over graphs," in *2022 IEEE Texas Power and Energy Conference (TPEC)*, 2022, pp. 1–6.

[22] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Advances in Neural Information Processing Systems*, 2018, pp. 539–548.

[23] S. Paul and S. Chowdhury, "A scalable graph learning approach to capacitated vehicle routing problem using capsule networks and attention mechanism," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 86236. American Society of Mechanical Engineers, 2022, p. V03BT03A045.

[24] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, "A note on learning algorithms for quadratic assignment with graph neural networks," *stat*, vol. 1050, p. 22, 2017.

[25] S. Paul, W. Li, B. Smyth, Y. Chen, Y. Gel, and S. Chowdhury, "Efficient planning of multi-robot collective transport using graph reinforcement learning with higher order topological abstraction," *arXiv preprint arXiv:2303.08933*, 2023.

[26] S. A. M. Shihab, P. Wei, J. Shi, and N. Yu, "Optimal evtol fleet dispatch for urban air mobility and power grid services," in *AIAA Aviation 2020 Forum*, 2020, p. 2906.

[27] Q. Wei, G. Nilsson, and S. Coogan, "Scheduling of urban air mobility services with limited landing capacity and uncertain travel times," in *2021 American Control Conference (ACC)*, 2021, pp. 1681–1686.

[28] P. K. Kumar, J. Witter, S. Paul, K. Dantu, and S. Chowdhury, "Graph learning based decision support for multi-aircraft take-off and landing at urban air mobility vertiports," in *AIAA SCITECH 2023 Forum*, 2023, p. 1848.

[29] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015.

[30] A. Behjat, H. Manjunatha, P. K. Kumar, A. Jani, L. Collins, P. Ghassemi, J. Distefano, D. Doermann, K. Dantu, E. Esfahani *et al.*, "Learning robot swarm tactics over complex adversarial environments," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021, pp. 83–91.

[31] C. Zeng, G. R. Hecht, P. K. Kumar, R. K. Shah, E. M. Botta, and S. Chowdhury, "Learning robust policies for generalized debris capture with an automated tether-net system," in *AIAA SCITECH 2022 Forum*, 2022, p. 2379.

[32] A. Sanders, *An introduction to Unreal engine 4*. AK Peters/CRC Press, 2016.

[33] A. Andreeva-Mori, "A study on finding a substitute to the first come-first served rule applied to aircraft sequencing," in *28th International Congress of the Aeronautical Sciences (ICAS)*, vol. 10, no. 2, 2012, p. 2012.