Graph Learning-based Fleet Scheduling for Urban Air Mobility under Operational Constraints, Varying Demand & Uncertainties

Steve Paul
Department of Mechanical and Aerospace Engineering, University at Buffalo
Buffalo, NY, USA
stevepau@buffalo.edu

Jhoel Witter
Department of Mechanical and Aerospace Engineering, University at Buffalo
Buffalo, NY, USA
jhoelwit@buffalo.edu

Souma Chowdhury
Department of Mechanical and Aerospace Engineering, University at Buffalo
Buffalo, NY, USA
soumacho@buffalo.edu

ABSTRACT

This paper develops a graph reinforcement learning approach to online planning of the schedule and destinations of electric aircraft that comprise an urban air mobility (UAM) fleet operating across multiple vertiports. This fleet scheduling problem is formulated to consider time-varying demand, constraints related to vertiport capacity, aircraft capacity and airspace safety guidelines, uncertainties related to take-off delay, weather-induced route closures, and unanticipated aircraft downtime. Collectively, such a formulation presents greater complexity, and potentially increased realism, than in existing UAM fleet planning implementations. To address these complexities, a new policy architecture is constructed, primary components of which include: graph capsule conv-nets for encoding vertiport and aircraft-fleet states both abstracted as graphs; transformer layers encoding time series information on demand and passenger fare; and a Multi-head Attention-based decoder that uses the encoded information to compute the probability of selecting each available destination for an aircraft. Trained with Proximal Policy Optimization, this policy architecture shows significantly better performance in terms of daily averaged profits on unseen test scenarios involving 8 vertiports and 40 aircraft, when compared to a random baseline and genetic algorithm-derived optimal solutions, while being nearly 1000 times faster in execution than the latter.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent planning**; **Planning under uncertainty**; **Sequential decision making**.

KEYWORDS

Multi-Agent Systems, Urban Air Mobility, Reinforcement Learning

ACM Reference Format:
Steve Paul, Jhoel Witter, and Souma Chowdhury. 2024. Graph Learning-based Fleet Scheduling for Urban Air Mobility under Operational Constraints, Varying Demand & Uncertainties . In *Proceedings of ACM SAC Conference (SAC'24).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3605098.3635976

SAC'24, April 8 –April 12, 2024, Avila, Spain

ACM ISBN 979-8-4007-0243-3/24/04…$15.00
https://doi.org/10.1145/3605098.3635976

1 INTRODUCTION

The concept of Urban Air Mobility (UAM) utilizes electric vertical take-off and landing (eVTOL) aircraft [2] to offer automated air transportation for passengers, cargo, and critical (e.g., air-ambulance) services, with a projected global market size of $1.5 trillion by 2040 [8, 10]. The economic viability of this new mode of transportation depends on the ability to operate a sufficiently large number of increasingly autonomous eVTOLs in any given market (i.e., achieve high penetration). This in turn demands safe airspace management and robust fleet planning solutions among others. More specifically, deploying a regional UAM network comprising scores of eVTOL aircraft requires an effective scheduling framework that can adapt to the unique demand patterns (that's different from general aviation) and aircraft and airspace constraints (distinct from other modes of regional/metropolitan transportation), while maximizing profitability and mitigating energy footprint [7]. These scheduling problems usually take the form of complex nonlinear Combinatorial Optimization (CO) problems, which can be addressed through classical optimization, heuristic search, and learning-based approaches. Approaches that provide local optimal solutions for small UAM fleet scheduling problems [13, 26, 31] often present computational complexity that makes them impractical for online decision-making. By taking a multi-agent task planning perspective of the online fleet scheduling problem, in this paper, we propose a new reinforcement learning (RL) based solution. Moreover, this approach accounts for important problem complexities and constraints that are otherwise often overlooked by existing methods. These include constraints related to airspace corridors, aircraft charging, vertiport capacity, weather-induced uncertainties, and time-varying demand [2]. Air corridors in UAM are designated routes or paths in the airspace that are specifically allocated for the operation of the eVTOL. Some of the associated technical challenges are summarized below.

Dynamic environment: The scheduling framework needs to consider real-time factors such as airspace and weather conditions, ground traffic, infrastructure availability, and demand uncertainty [2]. Thus, shorter time-scale, adaptive and robust planning is favored over fixed, deterministic, and/or day-ahead planning. It is also computationally challenging to resolve uncertainties in online planning.

Conflict resolution: The framework should facilitate sharing eVTOLs' state information and allow for trajectory and speed adjustments to ensure safe and optimal sharing of the airspace, which introduces additional constraints on decision-to-fly actions w.r.t. the route between any two vertiports [2].

Optimization & resource allocation: Optimizing (scarce) resource allocation, such as vertiport parking slots, charging stations, and air corridor capacity, is essential to prevent unnecessary delays in charging, takeoff, and other operations, which in turn impose additional constraints on scheduling actions [2].

Learning & adaptation: The capability of learning from past experiences, interactions, and feedback to adapt their decision-making strategies is needed. Learning algorithms facilitate agent adaptation to changing conditions, leading to improved efficiency and enhanced system performance by shifting the computational expense of training offline [33]. Similar challenging characteristics can be found in other fleet planning and multi-robot transport or fulfillment planning problems.

Note that some of the above-stated challenges also appear in other critical fleet planning, multi-robot transport and fulfillment planning problems. To address these stated problem complexities and provide efficient online-executable policies for fleet scheduling, we explore the use of specialized Graph Neural Networks (GNNs) [22, 23]. Our approach builds upon existing work in multi-robot task allocation, and through numerical experiments, demonstrates superior performance compared to standard RL-based and heuristic optimization-based solutions, as well as a feasible-random baseline.

Related Work: There is a small but growing body of work in UAM fleet planning, which has provided impetus for transitioning optimization and learning formalisms to advance this emerging concept. However, the majority of the existing work overlooks some of the important guidelines proposed by the US FAA [2] w.r.t. UAM airspace integration. For example, existing work usually lacks considerations for air corridors, range/battery capacity constraints, unforeseen events such as route closures due to bad weather or off-nominal events, or dysfunctional eVTOLs [5, 13, 21, 39]. Traditional methods such as Integer Linear Programming (ILP) and metaheuristics are not suitable for solving related NP-hard fleet scheduling problems in a time-efficient manner [12, 18, 19, 25, 28, 38, 40]. For perspective, here we consider hour-ahead planning, in order to enable enough capacity to adapt to varying demand and state of routes affected by weather and unanticipated aircraft downtime. Learning-based methods have in recent years shown promise for generating policies for CO problems with relatable characteristics [3, 9, 11, 15–17, 20, 23, 32, 34]. In their current form, however, they consider fewer complexities, tackle simpler problem scenarios or do not provide explicit capture of the problem-specific context.

We hypothesize that in order to address these complexities in UAM fleet scheduling or related problems, in a manner that would be both generalizable across scenarios/environments, a suitable representation of the problem space is needed. Subsequently, we need to identify a neural architecture that can efficiently operate on this representation to provide reliable solutions with a small optimality gap. To this end, firstly we explore the use of a graph abstraction of the eVTOL state and vertiport state space, which is amenable to adding or changing problem/environment features. Secondly, we create a lightweight simulation environment that incorporates the modeling of the principal constraints and uncertainties. Finally, we propose a new graph neural net (GNN) type policy architecture to operate on the graph abstraction of the problem and utilize the simulation environment to generate hour-ahead sequential actions for eVTOLs over a generic (12-hr) day of operation, acting in the role of a centralized planner. The policy network combines a Graph Capsule Convolutional Neural Network (GCAPCN) [37] to encode vertiport and eVTOL state information, a Transformer encoder network to incorporate time-series data on demand and fare (similar to [4]), and a feedforward network to encode passenger transportation cost. Additionally, a Multi-head Attention mechanism is used to fuse the encoded information and problem-specific context, to compute the sequential actions [15, 36, 37].

The Main Contributions of this paper can thus be summarized as **1)** Formulating the UAM fleet scheduling problem as a Markov Decision Process (MDP), and architecting a centralized encoder-decoder policy network, where the state of the UAM network (vertiports and aircrafts) is embedded by a special Graph Neural Network (GNN), with the demand, passenger fare, operating cost and air corridor availability information processed by different Context encoders and then concatenated. The scheduling actions are computed using a Multi-head Attention (MHA) based action decoder that is fed by the GNN and context. **2)** Integrating the encoder-decoder policy network with a new simulation environment that models the daily operation of 40 eVTOLs across 8 vertiports, associated uncertainties, and airspace/aircraft constraints, and provides data on demand, fare, and operational costs. **3)** Training the encoder-decoder network via policy gradient techniques and demonstrating its ability to generalize across unseen scenarios and uncertainties. *We expect that, with problem-specific design of the context portion, this policy architecture can generalize to a wider range of multi-agent/vehicle/robot scheduling problems with similar complex characteristics, namely graph-abstractable task/resource space, time-varying and uncertain environment properties, and the need for sequential actions that satisfy a large set of physical and operational constraints.*

Paper Outline: The next section describes the UAM fleet scheduling problem and its MDP formulation. Section 3 explains the proposed solution, including the state encoding and action decoding. Section 4 covers training and experimental evaluations, and Section 5 presents concluding remarks.

2 PROBLEM DESCRIPTION & FORMULATION

A UAM network includes vertiports and eVTOLs (or aircraft) for passenger transportation, with vertiports providing take-off/landing spots, charging spots, parking areas, and terminals for boarding and departing. We consider a concept of operations (ConOps), where every vertiport is connected to each other (i.e., a *fully* connected vertiport network [24]) by a route that comprises air corridors, as shown in Fig. 1. For safe operation, eVTOLS can only use these air corridors for flight. Since UAM is still an emerging concept, the exact connectivity structure of a UAM network is not yet established. Hence we assumed a fully connected network, i.e., where all the nodes are connected to each other. The corridors have regulations regarding the required distance gap for safe navigation. Here, we consider the corridors to be straight tube-shaped air columns. We consider there to be 4 air corridors between two vertiports with two corridors for each direction. The UAM network is defined as involving N vertiports, and N_K number of eVTOLs, with each eVTOL having a maximum passenger seating capacity of C (=4) and maximum battery capacity of $B_{\max}$ (=110kWh). Let V and K be the set of all vertiports and eVTOLs, respectively. Each vertiport $i \in V$

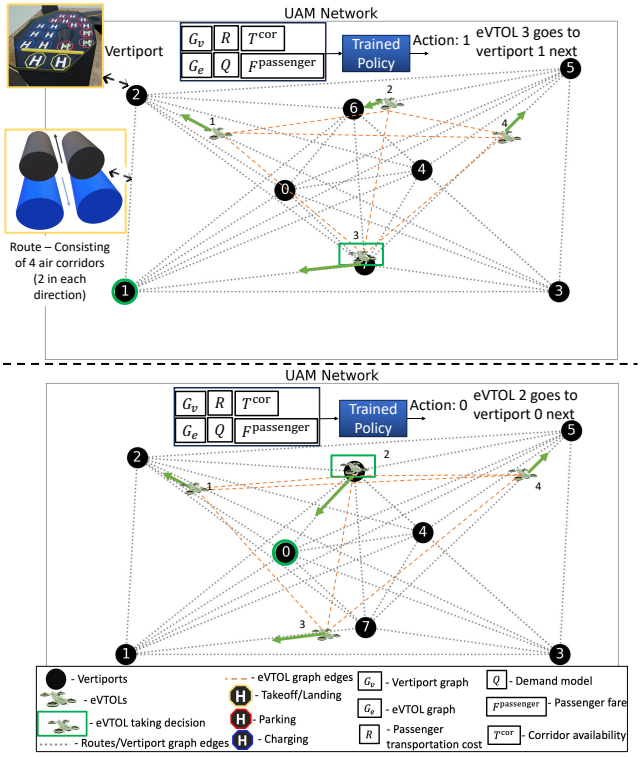

Figure 1: The trained policy implemented on the UAM network: The two images show two consecutive decision-making instances. For clarity of illustration, only 4 eVTOls are shown.

has a maximum number ($C^{\text{park}}_{\text{max}}$=10) of eVTOLs it can accommodate at a time and the total number of charging stations it contains ($C^{\text{charge}}_{\text{max}}$=6). Some of the vertiports that do not have a charging facility, called vertistops ($V_s \subset V$), are only used for landing/take-off, and passenger boarding and have parking spaces. For computing the cost of transportation, we define $R_{i,j}$ to be the cost of transporting a passenger from vertiport i to j. The travel demand between vertiports is modeled based on real data as explained in Section 2.1. Each vertiport $i \in V$, has an expected take-off delay T^{TOD}_i which will affect every take-off from the vertiport during an episode. Here, an episode refers to the UAM operation for a specific time period.

We make the following assumptions for setting up this problem: **1)** A single service provider runs the entire UAM network, and full observation of the states of aircraft and vertiports in the network is available to the central agent making the scheduling decisions, which is reasonable under current communication capabilities in urban areas, and given that planning occurs at 15-60 min time scales. **2)** Every eVTOL can commute between any two vertiports if it has enough battery charge (considering a safety margin) for the commute. **3)** The resistive loss of the batteries is negligible. **4)** The state information of eVTOLs and vertiports is always accessible for decision-making, which is crucial for ensuring passenger safety in aviation applications. **5)** An estimate of the probability of unplanned technical grounding of eVTOLs is available. **6)** While route closure on a given day of operation is not known apriori, an estimate of the probability of route closure (due to factors such as bad weather) is however available. **7)** The expected take-off delay ($T^{\text{TOD}}_i, i \in V$) at every vertiport can be estimated and is considered to be less than 6 minutes. We consider a probability of route closure between two vertiports as $P^{\text{closure}}_{i,j} (\leq 0.05), \forall\, i, j \in V$. Once the route is closed, there will not be any more commutes between these two vertiports during the rest of the episode. We consider the probability of an eVTOL to become dysfunctional as $P^{\text{fail}}_k (\leq .005), \forall\, k \in K$ during an episode. In every episode, we consider a randomly assigned take-off delay (of $<$ 30 mins) for any vertiport-i, based on a Gaussian distribution with mean T^{TOD}_i and a standard deviation of 6 minutes. We assume this probability distribution of take-off delay to be known prior to the scheduling. While these uncertainties are modeled for realism, and seeded with prescribed values due to lack of historical data in this regard, these values can be readily tweaked once data (or forecasts) become available. Passenger pricing for a journey is based on operational cost per passenger and the demand for the trip. Upon reaching a vertiport, each eVTOL is fully charged before proceeding to the next destination (excluding vertistops) or before being parked in an available spot if it is to be idle.

The planning objective is to maximize the daily profit by optimizing the schedule of eVTOL flights between vertiports to meet travel demand. Each decision-making instance involves assigning an eVTOL (currently at a vertiport) to another vertiport to fly to, or instructing it to remain idle for another 15 mins. Factors such as demand, battery charge, and operational costs are taken into account. The current implementation focuses on 4-hour-long episodes (for computational ease of training) between 6 am to 6 pm without an end-of-episode constraint. Each episode is independent, starting with a random number of fully charged eVTOLs at each vertiport, subject to vertiport capacity constraints. The passenger demand model, eVTOL battery model, and optimization formulation of the problem are further discussed below.

2.1 Demand Model

Passenger demand modeling is used to generate stochastic passenger requests between different vertiports. The demand is based on forecasted data from [35], and we assume to know the expected demand for each hour during daily operating hours. The demand between two vertiports i and j at a time t, $Q(i, j, t)$, mimics the demand patterns of a major city's subway system, where certain stations close to commercial hubs and workplaces experience higher traffic. In our scenario, a subset of vertiports, $V_B \subset V$, is designated as high-demand vertiports. The demand between vertiports in V_B is higher compared to those in $V - V_B$. We consider two peak hours: 8.00-9.00 am (T^{peak1}) and 4.00-5.00 pm (T^{peak2}). Vertiports in V_B experience peak demand during both hours, resembling the morning and evening rush hours for commuting between home and workplace. This represents high demands from vertiports in $V - V_B$ to those in V_B during T^{peak1}, and vice versa in T^{peak_2}.

2.2 eVTOL model

The eVTOL vehicle model considered here is the same as in [21] (City Airbus eVTOL aircraft), having a maximum cruise speed of 74.5 mph, maximum passenger capacity of 4, and a maximum range of 50 miles The operating cost of the vehicle is about $0.64 per mile

[31]. We also assume that for every eVTOL k, a downtime probability of being dysfunctional is given by P_k^{fail}; this is based on work on aircraft predictive maintenance [29]. Once an eVTOL become dysfunctional, it can no longer be in service for the remainder of the episode. The battery model is considered to be the same as that in [21], which consists of a maximum capacity of $B_{\max}$= 110 kWh. If B_t^k is the battery charge of eVTOL k at time t, and assuming the eVTOL travels from vertiport i to j, the charge for the next time step B_{t+1}^k can be computed as $B_{t+1}^k = B_t^k - B_{i,j}^{\text{charge}}$ Here $B_{i,j}^{\text{charge}}$ is the charge required to traverse between vertiports i and j. The batteries are charged at vertiports with a charging rate of 150 kW.

2.3 Passenger Fare & Electricity Pricing Models

The passenger price consists of a fixed base fare, F^{base}, of \$5, and a variable fare, $F^{\text{passenger}}$. The variable fare between two vertiports for a passenger at a time t is computed as a function of the demand profile, $Q(i,j,t)$, and the operational cost, $R_{i,j}$ (between i and j), expressed as $F_{i,j,t}^{\text{passenger}} = R_{i,j} \times Q_{\text{factor}}(i,j,t)$. Here $Q_{\text{factor}}(i,j,t) = \max(\log(Q(i,j,t)/10), 1)$. This is a hand-crafted function that accounts for demand in passenger pricing. A constant electricity pricing, $\text{Price}^{\text{elec}}$ of \$0.2/kWh is used here [1].

2.4 Optimization Formulation

The objective function to be maximized in the fleet scheduling problem is the daily profit, given by the difference of the earned revenue, and the operating and electric-charging costs. Uncertainties due to route closures, eVTOL malfunctions, and expected delays affect this objective function. Here, the decision variables are the destination vertiports ($V_{k,l}^{\text{end}}$ in the following paragraph) of all the eVTOLs for all journeys. Hence, we are presented with a stochastic Integer Nonlinear Programming (INLP) problem.

Consider a time period T (with $|T|$ hours), with a start time of T^{start} and end time of T^{end}. We consider cases where $T^{\text{end}} <$ 6.00 pm, and $T^{\text{start}} = T^{\text{end}} - |T|$, and $T^{\text{start}} \geq$ 6.00 am. For every eVTOL, $k \in V_e$, let S_k^{jour} be the set of journeys taken during time period of T; this could be of different length for different eVTOLs. Let $N_{i,l}^{\text{passengers}}$ be number of passengers transported during the trip, $l \in S_k^{\text{jour}}$ by eVTOL k. Let B_l^k be the battery charge of eVTOL k just before its l'th journey, $V_{k,l}^{\text{start}}$ and $V_{k,l}^{\text{end}}$ be the respective start and end vertiports of eVTOL k during its l'th journey, $T_{k,l}^{\text{takeoff}}$ and $T_{k,l}^{\text{landing}}$ be the corresponding takeoff time and landing time.

Therefore, the total cost of transportation, ($\text{Cost}^{\text{Oper}}$) and charging, ($\text{Cost}^{\text{Elec}}$), and the revenue during T are given by:

$$\text{Cost}^{\text{Oper}} = \sum_{k \in K} \sum_{l \in S_k^{\text{jour}}} N_{i,l}^{\text{passengers}} \times R_{i,j},\ i = V_{k,l}^{\text{start}},\ j = V_{k,l}^{\text{end}} \quad (1)$$

$$\text{Cost}^{\text{Elec}} = \sum_{k \in K} \sum_{l \in S_k^{\text{jour}}} \text{Price}^{\text{elec}} \times B_{i,j}^{\text{charge}},\ i = V_{k,l}^{\text{start}},\ j = V_{k,l}^{\text{end}} \quad (2)$$

$$\text{Revenue} = \sum_{k \in K} \sum_{l \in S_k^{\text{jour}}} N_{i,l}^{\text{passengers}} \times F_{i,j,t}^{\text{passenger}}, \quad (3)$$
$$i = V_{k,l}^{\text{start}},\ j = V_{k,l}^{\text{end}},\ t = T_{k,l}^{\text{takeoff}}$$

Therefore, the objective function can be expressed as:

$$\max z = \text{Revenue} - \text{Cost}^{\text{Oper}} - \text{Cost}^{\text{Elec}} \quad (4)$$

We must also satisfy the following operational constraints:

$$N_{i,l}^{\text{passengers}} = \min(C, Q_{\text{act}}(i,j,t)), i = V_{k,l}^{\text{start}},\ j = V_{k,l}^{\text{end}},$$
$$t = T_{k,l}^{\text{takeoff}}\ \forall\ l \in S_k^{\text{jour}} \quad (5)$$

$$B_{T_{k,l}^{\text{takeoff}}}^k > B_{i,j}^{\text{charge}},\ i = V_{k,l}^{\text{start}},\ j = V_{k,l}^{\text{end}},\ \forall\ k \in K,\ \forall\ l \in S_k^{\text{jour}} \quad (6)$$

$$C_{i,t}^{\text{park}} \leq C_{\max}^{\text{park}},\ \forall\ i \in V_v,\ \forall\ t \in \left[T^{\text{start}}, T^{\text{end}}\right] \quad (7)$$

$$V_{l,k}^{\text{end}} \neq i,\ \text{if } A^{i,V_{l,k}^{\text{end}}} = 0\ \forall\ i \in V_v,\ \forall\ k \in K,\ \forall\ l \in S_k^{\text{jour}} \quad (8)$$

Here $B_{i,j}^{\text{charge}}$ is the amount of charge required for an eVTOL to fly from vertiport i to j, $Q_{\text{act}}(i,j,t)$ is the actual demand sampled from Q, P^{closure} represents the probability matrix for route closure, and A is a binary matrix such that $A_{i,j} = A_{j,i} = 0$, if the route between vertiports i and j is closed. Equations 1, 2, and 3 are used to compute the objective function (Eq. 4). Equation 5 is used to compute the number of passengers transported by eVTOL k on its l'th journey. Constraint in Eq. 6 ensures that eVTOL k has enough charge before taking off for its l'th journey. Constraint in Eq. 7 ensures that the number of eVTOLs parked in every vertiport does not exceed its maximum capacity. Constraint in Eq. 8 ensures that infeasible vertiports are not chosen during a decision-making instance. Later on, a Genetic Algorithm is applied on this exact optimization formulation to compute optimal solutions for comparisons with the learning based solutions.

2.5 MDP Formulation

In this work, we formulate the fleet scheduling problem as an MDP, where actions are computed sequentially for each eVTOL during its decision-making instance ($t \in [T^{\text{start}}, T^{\text{end}}]$). At each time step, an action is assigned to each eVTOL based on the current state of the vertiport network, which contains all necessary information for decision-making. Unlike other multi-agent approaches in smaller UAM settings, here we impose high safety standards [2]. Therefore, full state information is essential for decision-making, leading us to adopt a centralized decision-making scheme.

Graph Formulation for UAM vertiport Network: The UAM vertiport network is expressed as a graph, $G_v = (V_v, E_v, A_v)$, where V_v (=V) represents the set of nodes or vertiports; in this case, E_v represents the set of edges between the nodes, and A_v represents the adjacency matrix of the nodes. Since we consider a route closure probability P^{closure}, we compute the weighted adjacency matrix as $A_v = (\mathbf{1}_{N \times N} - P^{\text{closure}}) \times A$, where A is a matrix representing the route closure such that if $A_{i,j}$ is the route between nodes i and j ($i, j \in V_v$), then $A_{i,j} = A^{j,i} = 0$, if the route is closed; else it is equal to 1. Here, the properties, δ_i^t, of each node $i \in V_v$ at the time step t are: **1)** the x-y coordinates of the node/vertiport (x_i, y_i), **2)** the number of eVTOLs that are parked at vertiport i at time t, $C_{i,t}^{\text{park}}$, **3)** the earliest time at which a charging station is available T_i^{charge}, **4)** the expected take-off delay T_i^{TOD}, and **5)** a binary number I_i^{vstop} which takes a value of 1 if the node is a vertistop. Hence, $\delta_i^t = \left[x_i, y_i, C_{i,t}^{\text{park}}, T_i^{\text{charge}}, T_i^{\text{TOD}}, I_i^{\text{vstop}}\right]$, $\delta_i^t \in \mathbb{R}^6$.

Graph Formulation for eVTOLs Network: The state of eVTOLs is also represented as a graph, $G_e = (V_e, E_e, A_e)$, where V_e represents the set of eVTOLs, E_e represents the set of edges between the nodes, and A_e represents the adjacency matrix. We consider G_e to be fully connected. Each node $k \in V_e$ has its time-varying node properties, ψ_k^t. The properties of a eVTOL node are **1)** the coordinates of the destination vertiport (x_k^d, y_k^d), **2)** the current battery level as a fraction B_t^k, **3)** the next flight time T_k^{flight}, **4)** the next decision-making time T_k^{dec}, and **5)** the probability of failure P_k^{fail}. Therefore, $\psi_k^t = \left[x_k^d, y_k^d, B_t^k, T_k^{\text{flight}}, T_k^{\text{dec}}, P_k^{\text{fail}}\right]$, $\psi_i^t \in \mathbb{R}^6$.

State Space: The state information that will be used for computing the action at a time step t consists of: **1)** Vertiport graph G_v, **2)** eVTOLs graph G_e, **3)** Demand profile Q, **4)** Passenger fare $F^{\text{passenger}}$, **5)** Cost of per passenger transportation R, and **6)** Time at which it's safe to launch an eVTOL to the corridors $T^{\text{cor}} \in \mathbb{R}^{N\times N\times 2}$. It should be noted some of the state variable updates such as $T^{\text{flight}}, T^{\text{dec}}, T_i^{\text{charge}}, T_i^{\text{TOD}}$, etc. are not explicitly presented in this paper due to space constraints; however, they have been appropriately implemented programmatically in the developed RL environment. To handle the large state space, we use a Graph Neural Network (GNN) to compute a fixed-length feature vector that represents the information of vertiports and eVTOLs. The demand profile and passenger fare are modeled as time-series data and extracted by a Transformer encoder. The cost of per-passenger transportation is represented by a learned feature vector from a feedforward network. Corridor closure information T^{corr} is flattened and concatenated with the aforementioned embeddings.

Action Space: During each decision-making instance, there will be one eVTOL (or aircraft) that will be deciding its next destination, i.e., select an action, $V_{k,l}^{\text{end}} \in V_v$), where V_v includes all available vertiports. If it chooses the vertiport where it is currently located, it will wait or idle for 15 mins before triggering a new decision instance. We also consider a masking mechanism to prevent the selection of infeasible vertiport journeys, i.e., ones that violate any of the constraints presented in Eqs. 5, 6, and 7.

Reward: We consider a delayed reward strategy, where the total reward computed at the end of the episode is the ratio of the profit (Eq. 4) to the maximum possible episodic profit, i.e., $\sum_{i\in V, j\in V, t\in[T^{\text{start}}, T^{\text{end}}]}(Q(i,j,t) \times F_{i,j,t}^{\text{passenger}})$.

State Transition: Uncertain route closure and take-off delay, and uncertain variations in demand lead to a stochastic state transition here, which is computed by the simulation.

3 POLICY ARCHITECTURE & LEARNING FRAMEWORK

We develop an RL framework to compute actions for individual eVTOLs during decision-making instances. The policy model, as illustrated in Fig. 2, is called when a fully charged eVTOL is waiting to be assigned a destination. If an eVTOL remains grounded due to route closures, destination crowding, or insufficient battery, the policy model is re-queried after a 15-min wait. The policy model takes the state as input and outputs the probability of selecting the next destination for the eVTOL. As shown in Fig. 2), it combines GNN-based and Transformer-based encoders, a feedforward network,

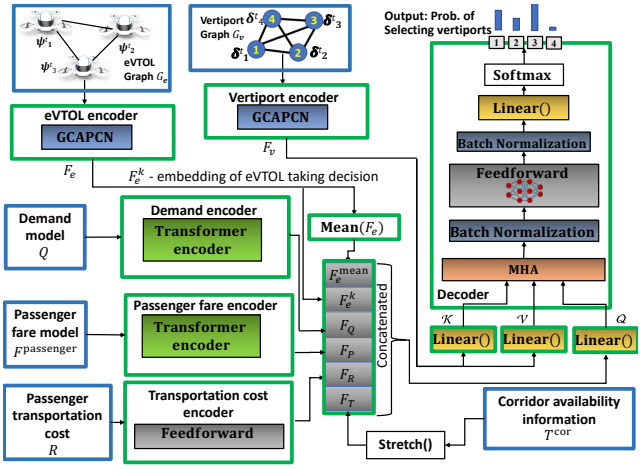

Figure 2: The CapTAIN policy network consists of GCAPCN and Transformer encoders, and a decoder. Green and blue blocks represent the policy and the state space, respectively.

and a Multi-head Attention (MHA) based decoder. The state information is encoded as fixed-length (of l_{embed}) feature vectors using encoders and a context module. These vectors are used by the decoder to compute actions sequentially. We call the proposed policy network as **Cap**sule **T**ransformer **A**ttention-mechanism **I**ntegrated **N**etwork (**CapTAIN**). Each component of this policy model is further explained below.

3.1 State Encoding

The state information consists of the information of UAM vertiports, eVTOLs, passenger demand, passenger fare, passenger transportation costs, and electricity pricing. This section describes how the state information is encoded.

3.1.1 Vertiport and eVTOL State Encoding with GNN:. We use a GNN to compute feature vectors for the vertiport and eVTOL state information, both of which are abstracted as graphs. Specifically, we employ a Graph Capsule Convolutional Neural Network (GCAPCN) for learning local and global structures with node properties (Fig. 2 top-left). The node embedding in GCAPCN is permutation invariant, similar to the approach described in [21]. GCAPCN is a GNN introduced in [37] to address the limitations of Graph Convolutional Neural Networks (GCN), enabling the encoding of global information using capsule networks as presented in [6]. Further description of the GCAPCN architecture can be found in [21]. We use GCAPCN to compute node embeddings for the vertiport and eVTOL graphs, $F_e \in \mathbb{R}^{N_K \times l_{\text{embed}}}$ and $F_v \in \mathbb{R}^{N\times l_{\text{embed}}}$ respectively, where l_{embed} is the embedding length.

3.1.2 Transformer-based Encoding of Demand & Passenger Fare: Here, we utilize a Transformer Architecture to compute learnable embeddings for entities that can be represented as time-series data, such as demand distribution over time and passenger fare (Fig. 2 bottom-left). The Transformer architecture, originally introduced in [36] and widely used in applications such as Natural Language Processing [36] and time series forecasting [41], follows an encoder-decoder approach based on self-attention. The encoder maps an

input sequence to continuous representations, and the decoder generates an output sequence by attending to relevant information. In our work, we utilize the Transformer encoder to compute continuous and learnable feature vectors for time-series data. The forecasted demand between vertiports during each hour (Q) and the passenger fare ($F^{\text{passenger}}$) are processed by two separate Transformer networks into learned feature vectors, $F_Q \in \mathbb{R}^{l_{\text{embed}}}$ and $F_P \in \mathbb{R}^{l_{\text{embed}}}$, respectively.

3.1.3 Passenger transportation cost encoding: The passenger transportation information (R) can be considered as a matrix of size $N \times N$. This information can be encoded as a feature vector F_R of length l_{embed} by passing through a feedforward (FF) network.

3.1.4 Corridor Availability encoding: For every corridor, we keep track of the time at which it is safe for a new eVTOL to enter the corridor (Fig. 1) subject to minimum separation requirements governed by safety. The corridor availability T^{cor}, a tensor of size $N \times N \times 2$. T^{cor} is stretched into a 1-D vector F_T of length $N \times N \times 2$, as shown in Fig. 2 bottom-right.

3.1.5 Context: The context vector is computed by taking the linear transformation of a concatenated vector of the mean of the eVTOL node embedding (F_e) given by the GCAPCN encoder, the embedding of the eVTOL taking decision (F_e^k), demand and passenger fare encoding given by the transformers (F_Q and F_P), the transportation cost encoding (F_R) given by the FF, and the corridor availability vector (F_T).

3.2 Action Decoding

We use a Multi-head Attention-based decoder as shown in Fig. 2 top-right, to compute which vertiport to visit next, given the vertiport node embeddings (F_v) and the Context vector. The attention mechanism computes compatibility scores between the context and the node embeddings, selecting the destination for the eVTOL based on the highest compatibility score. The choice of the current vertiport itself indicates a stay-idle decision.

3.3 Simulation Environment & Training Algorithm

We considered a hypothetical area of 50 x 50 sq miles with 8 vertiport locations (Fig. 1), including 2 vertistops, V_s (1 and 6). Four vertiports (0, 4, 6 and 7) are designated as high-traffic (V_B) due to their high trip demand. The location of vertiports remained the same for each training scenario, while the hourly demand values changed for each episode. The simulation environment is implemented in Python using the OpenAI Gym interface, making it compatible with standard RL training algorithms such as A2C [14] and PPO [30] through the stable-baselines3 library. To train the policy network, here we use PPO from **stable-baselines3** [27]. A 2-layer neural network with input and intermediate length l_{embed} and LeakyReLU activation is used as the value network.

4 EXPERIMENTAL EVALUATION

In order to assess the importance of each principal component of the proposed CapTAIN policy model, e.g., the novel encoder choices, we train two alternate policy models, using two GPUs (NVIDIA Tesla V100, 16GB RAM). The first of these alternate models ablates the two GCAPCN encoders by replacing it with a Multi-Layered Perceptron (MLP) and is named the **Ablation-GCAPCN**, while the second one ablates the Transformer network by replacing it with an MLP, and is named the **Ablation-Transformer**. These alternate models are also compared against two other baselines (Section 4.2).

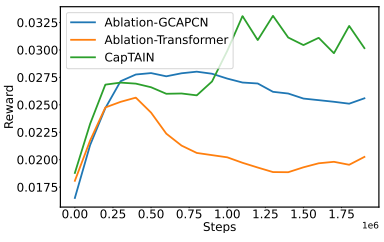

Figure 3: Convergence plot for all the trained models

Details	Values
Total steps	2e6
Rollout size	20000
Optimizer	*Adam*
Learning step	1e-7
Entropy coef	0.01
Value coef	0.5

Table 1: Training settings

The first baseline is given by a **standard elitist Genetic Algorithm (GA)** that uses a population size of 30, max iteration of 30, mutation probability of 0.1, elite ratio of 0.1, and crossover probability of 0.5. Here, the GA is implemented in batches of 30 sequential decisions. During each batch, the optimization variables are the 30 decisions where each decision takes an integer value between 1 and N (for each vertiport), and the objective function is computed using Eq. 4 at the end of the 30 decisions. We take the 30 decision variables and run the simulation with these 30 decisions implemented sequentially resulting in an updated environment state. The optimization of the next 30 decisions starts with the current state of the environment. This is continued until the end of an episode. A standard Python package[1] is used to implement the GA.

The second baseline method is a **Feasibility Preserving Random Walk (Feas-RND)** approach that randomly selects a vertiport from the set of feasible choices (satisfying Eqs. 5,6, 7, and 8) as the next destination during each decision-making instance.

Here onward, the Ablation-GCAPCN and Ablation-Transformer policy models are called the two learning-based baselines, while the GA and Feas-RND are called the two non-learning-based baselines.

4.1 Training & Convergence

The three policies are trained using PPO for 2 million steps based on the parameters in Table 1. From the convergence plots in Fig. 3, it can be seen that CapTAIN converges to a higher episodic mean reward compared to the other two policies. Ablation-Transformer episodic rewards improved until 900K steps, after which the performance started to deteriorate. This demonstrates the utility of the combination of the two encoder components (GCAPCN and Transformer) to provide better learning capability. Next, these trained models are tested against the non-learning-based methods (GA and Feas-RND), with the latter implemented on a 2.6 GHz Intel core i7 MacOS 11.2.3 system.

4.2 Generalizability & Ablation Study

The trained RL models and baselines are tested on 100 unseen episodes, which are each defined as a 12 hr (6.00 am–6.00 pm) operation of the UAM network. Figure 4 compares the results across all 100 scenarios based on the mean episodic reward and episodic

[1] https://pypi.org/project/geneticalgorithm/

profit. Figure 5 provides further comparisons in terms of the following metrics: total number stay-idle decisions, total number of flight decisions, total idle time, total flight time and total delay, computed across all eVTOLs over entire training episodes. It is observed

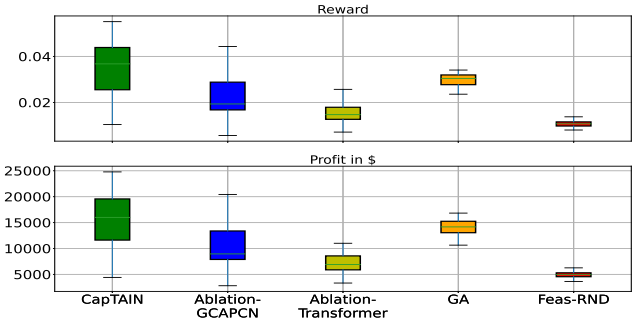

Figure 4: Comparative analysis of the 5 methods on total episodic reward, total episodic profit, total idle decisions, and total flight decisions.

from Fig. 4 that CapTAIN outperforms GA in terms of the average episodic rewards, achieving a mean reward of 0.34 vs. the 0.29 mean reward obtained by GA. To assess the significance of this difference, we perform a statistical T-test, with the null hypothesis being that both methods' mean performance is the same. The p-value is found to be $6{\times}10^{-6}(< .05)$, which indicates that CapTAIN has a statistically significant performance advantage over GA. As expected, Feas-RND performed the worst.

As seen from Fig. 4, the comparative trend (or ranking in terms) of the total episodic profit is very similar to that the episodic reward across the five methods tested, with CapTAIN performing the best. It is also notable that the Ablation-Transformer performs worse than Ablation-GCAPCN. This shows that the Transformer encoder contributes more strongly to generalizability compared to GCAPCN, at least under the current problem settings.

4.2.1 Computation Time analysis. The solution computing time for the GA is determined by adding the total time for which the optimization (in batches) is performed. Similarly for CapTAIN and the other learning-based methods, the total episodic computing time is the sum of the computing times for the forward propagation through the policy network throughout an episode. It's found that the average episodic computation time required by the GA is about 1,774 seconds, while for CapTAIN, it is only 2.1 seconds.

4.3 Further Analysis of Decision-Making

In order to physically interpret the diverse nature of the decision-making by the different methods, we track the total number of idle decisions, the total number of flight decisions, and the total episodic flight time, idle time and delays per eVTOL. From Fig. 5, it can be seen that Feas-RND and GA have comparatively more number of flight decisions and fewer idle decisions, and consequently higher episodic flight time per eVTOL and lower episodic idle time per eVTOL (Fig. 5), compared to CapTAIN. Interestingly, these observations show that more flights do not necessarily result in greater overall profits. This phenomenon is caused by the difference in demand and pricing across peak (higher fares) and off-peak (lower fares) hours. Here CapTAIN is taking advantage of this difference to provide better trade-offs in a number of peak/off-peak flights, leading to more favorable profit generation. Both GA and Feas-RND are also observed from Fig. 5 to result in lower total episodic delay per eVTOL, compared to CapTAIN. This shows that CapTAIN compromises delays with the objective of a higher profit, which could also be an artifact of delays not directly affecting demand or fares.

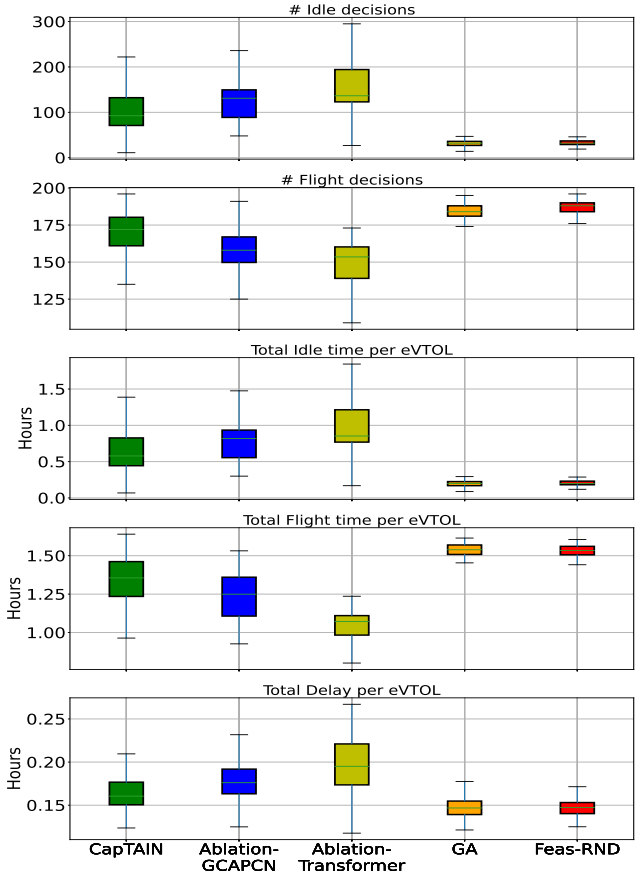

Figure 5: Comparing the effect of the 5 methods on total idle time, total flight time, and total delay time per eVTOL.

5 CONCLUSIONS

We proposed a graph RL approach using a new encoder-decoder policy architecture to perform UAM fleet scheduling, one that uniquely considers uncertainties (due to delays, aircraft downtime, and route closures), time-varying demand, vertiport constraints, and airspace constraints. The state of eVTOL aircraft and vertiport occupancy were expressed as graphs, while demand forecast and fares were treated as time-series data. Our novel architecture (CapTAIN) comprises Graph Capsule Convolutional Networks, Transformer encoders, feedforward context layers, and a Multi-head Attention-based decoder to compute sequential actions. Each of these components of CapTAIN is designed to play a specific role that caters to a specific complexity of the fleet scheduling problem, such as generalizable embedding of the structural information of the vertiports and eVTOL graphs, and transforming time-series data for demand and passenger fare into context vectors. The policy was trained using PPO and evaluated on 100 unseen scenarios. Compared to non-learning-based methods (GA and Feas-RND), our CapTAIN achieved better performance with up to 3 orders of magnitude faster

computation time than GA. Ablation studies showed that the Transformer encoder had a greater impact on performance than graph neural net encoders. Future directions include extending planning horizons and incorporating end-of-episode constraints to alleviate the limitations of the current myopic training implementation of the policy. Decentralized policies suitable for more realistic scenarios with multiple stakeholders operating vertiports and eVTOL fleets can also be explored in the future.

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research (ONR) award N00014-21-1-2530 and the 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 ONR or the NSF.

REFERENCES

[1] [n. d.]. See electric rates available to your home/business (updated today):. https://www.electricchoice.com/electricity-prices-by-state/

[2] 2023. Urban Air Mobility (UAM) Concept of Operations -v2.0. https://www.faa.gov/sites/faa.gov/files/Urban%20Air%20Mobility%20%28UAM%29%20Concept%20of%20Operations%202.0_0.pdf

[3] Thomas D Barrett, William R Clements, Jakob N Foerster, and Alex I Lvovsky. 2019. Exploratory combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1909.04063* (2019).

[4] Dung David Chuwang and Weiya Chen. 2022. Forecasting daily and weekly passenger demand for urban rail transit stations based on a time series model approach. *Forecasting* 4, 4 (2022), 904–924.

[5] Malintha Fernando, Ransalu Senanayake, Heeyoul Choi, and Martin Swany. 2023. Graph Attention Multi-Agent Fleet Autonomy for Advanced Air Mobility. *arXiv preprint arXiv:2302.07337* (2023).

[6] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming Auto-Encoders. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 44–51.

[7] Shyh In Hwang and Sheng Tzong Cheng. 2001. Combinatorial Optimization in Real-Time Scheduling: Theory and Algorithms. *Journal of Combinatorial Optimization* (2001). https://doi.org/10.1023/A:1011449311477

[8] Woodrow Bellamy III. [n. d.]. Evtol investments will continue billion dollar trend in 2021. http://interactive.aviationtoday.com/avionicsmagazine/february-march-2021/evtol-investments-will-continue-billion-dollar-trend-in-2021/

[9] Roshni Anna Jacob, Steve Paul, Wenyuan Li, Souma Chowdhury, Yulia R. Gel, and Jie Zhang. 2022. Reconfiguring Unbalanced Distribution Networks using Reinforcement Learning over Graphs. In *2022 IEEE Texas Power and Energy Conference (TPEC)*. 1–6. https://doi.org/10.1109/TPEC54980.2022.9750805

[10] Adam Jonas, Adam Jonas, Head of Global Auto, and Shared Mobility Research. [n. d.]. Are flying cars preparing for takeoff? https://www.morganstanley.com/ideas/autonomous-aircraft

[11] Yoav Kaempfer and Lior Wolf. 2018. Learning the Multiple Traveling Salesmen Problem with Permutation Invariant Pooling Networks. *ArXiv* abs/1803.09621 (2018).

[12] Nitin Kamra and Nora Ayanian. 2015. A mixed integer programming model for timed deliveries in multirobot systems. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 612–617.

[13] Sang Hyun Kim. 2020. Receding Horizon Scheduling of On-Demand Urban Air Mobility With Heterogeneous Fleet. *IEEE Trans. Aerospace Electron. Systems* 56, 4 (2020), 2751–2761. https://doi.org/10.1109/TAES.2019.2953417

[14] V. Konda and J. Tsitsiklis. 2003. OnActor-Critic Algorithms. *SIAM J. Control. Optim.* 42 (2003), 1143–1166.

[15] Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Attention, learn to solve routing problems!. In *7th International Conference on Learning Representations, ICLR 2019*. arXiv:1803.08475

[16] Prajit KrisshnaKumar, Jhoel Witter, Steve Paul, Hanvit Cho, Karthik Dantu, and Souma Chowdhury. 2023. Fast Decision Support for Air Traffic Management at Urban Air Mobility Vertiports using Graph Learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1580–1585.

[17] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*. 539–548.

[18] Clair E Miller, Albert W Tucker, and Richard A Zemlin. 1960. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)* 7, 4 (1960), 326–329.

[19] H. Mühlenbein. 1991. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Parallelism, Learning, Evolution*, J. D. Becker, I. Eisele, and F. W. Mündemann (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 398–406.

[20] Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. 2017. A note on learning algorithms for quadratic assignment with graph neural networks. *stat* 1050 (2017), 22.

[21] Steve Paul and Souma Chowdhury. 2022. A Graph-based Reinforcement Learning Framework for Urban Air Mobility Fleet Scheduling. In *AIAA AVIATION 2022 Forum*. 3911.

[22] Steve Paul and Souma Chowdhury. 2022. 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, V03BT03A045.

[23] Steve Paul, Payam Ghassemi, and Souma Chowdhury. 2022. Learning Scalable Policies over Graphs for Multi-Robot Task Allocation using Capsule Attention Networks. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 8815–8822.

[24] Xin Peng, Vishwanath Bulusu, and Raja Sengupta. 2022. Hierarchical Vertiport Network Design for On-Demand Multi-modal Urban Air Mobility. In *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*. 1–8. https://doi.org/10.1109/DASC55683.2022.9925782

[25] Yun Peng, Byron Choi, and Jianliang Xu. 2021. Graph Learning for Combinatorial Optimization: A Survey of State-of-the-Art. *Data Science and Engineering* (2021), 1–23.

[26] Priyank Pradeep and Peng Wei. 2018. Heuristic Approach for Arrival Sequencing and Scheduling for eVTOL Aircraft in On-Demand Urban Air Mobility. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. 1–7. https://doi.org/10.1109/DASC.2018.8569225

[27] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[28] A E Rizzoli, R Montemanni, E Lucibello, and L M Gambardella. 2007. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence* 1, 2 (2007), 135–151. https://doi.org/10.1007/s11721-007-0005-x

[29] Marc Josef Schoppmann. 2022. *The operation of eVTOLs in the urban air mobility sector: use case & operator assessment*. Ph. D. Dissertation.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[31] Syed Arbab Mohd Shihab, Peng Wei, Jie Shi, and Nanpeng Yu. 2020. Optimal eVTOL Fleet Dispatch for Urban Air Mobility and Power Grid Services. *AIAA AVIATION 2020 FORUM* (2020).

[32] Quinlan Sykora, Mengye Ren, and Raquel Urtasun. 2020. Multi-agent routing value iteration network. In *37th International Conference on Machine Learning, ICML 2020*. arXiv:2007.05096

[33] David P Thipphavong, Rafael Apaza, Bryan Barmore, Vernol Battiste, Barbara Burian, Quang Dao, Michael Feary, Susie Go, Kenneth H Goodrich, Jeffrey Homola, et al. 2018. Urban air mobility airspace integration concepts and considerations. In *2018 Aviation Technology, Integration, and Operations Conference*. 3676.

[34] Ekaterina V. Tolstaya, James Paulos, Vijay R. Kumar, and Alejandro Ribeiro. 2020. Multi-Robot Coverage and Exploration using Spatial Graph Neural Networks. *ArXiv* abs/2011.01119 (2020).

[35] United States. Federal Highway Administration (Ed.). 2010. Our Nation's Highways 2010. FHWA-PL-10-023 (Jan. 2010). https://rosap.ntl.bts.gov/view/dot/904

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[37] Saurabh Verma and Zhi Li Zhang. 2018. Graph capsule convolutional neural networks. arXiv:1805.08090

[38] Xinyu Wang, Tsan-Ming Choi, Haikuo Liu, and Xiaohang Yue. 2016. Novel Ant Colony Optimization Methods for Simplifying Solution Construction in Vehicle Routing Problems. *IEEE Transactions on Intelligent Transportation Systems* 17, 11 (2016), 3132–3141. https://doi.org/10.1109/TITS.2016.2542264

[39] Qinshuang Wei, Gustav Nilsson, and Samuel Coogan. 2021. Scheduling of Urban Air Mobility Services with Limited Landing Capacity and Uncertain Travel Times. In *2021 American Control Conference (ACC)*. 1681–1686. https://doi.org/10.23919/ACC50511.2021.9482700

[40] Tiehua Zhang, WA Gruver, and Michael H Smith. 1999. Team scheduling by genetic search. In *Intelligent Processing and Manufacturing of Materials, 1999. IPMM'99. Proceedings of the Second International Conference on*, Vol. 2. IEEE, 839–844.

[41] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.