A Graph-based Reinforcement Learning Framework for Urban Air Mobility Fleet Scheduling

Steve Paul* and Souma Chowdhury[†]
University at Buffalo, Buffalo, NY, 14260

Optimal scheduling of the fleet of aircraft comprising an urban air mobility (UAM) network is key to economically viable and sustainable integration of UAM networks within our existing urban and suburban transportation ecosystems. To this end, this paper firstly formulates the UAM fleet scheduling problem as a Markov Decision Process (MDP) over a graph space, with the graph representing the network of vertiports (and their dynamic properties, e.g., demand) being served by these aircraft. A simulation environment that incorporates real-world constraints associated with aircraft characteristics (e.g., max speed and battery capacity), passenger transport demand and electricity pricing is developed and used to evaluate schedules modeled by this MDP. The event-triggered action of each aircraft is determined in a decentralized manner using a novel policy model embodied by a neural network comprising a Graph Neural Network (GNN) based encoder and a Multi-head attention (MHA) based decoder. A policy gradient based reinforcement learning (RL) method is used to train this model. Motivated by the emerging work in learning to solve combinatorial optimization problems, this GNN-based policy model is expected to capture the local and global structural information of the UAM network, allowing the trained policies to generalize across demand and aircraft initialization scenarios. Compared to a simple feasible randomized baseline and a typical multi-layer neural network based policy, our method demonstrates a remarkable 25% better performance in terms of the estimated average daily profit.

Keywords: Fleet scheduling, Graph neural network, Reinforcement learning, Urban Air Mobility (UAM)

I. Nomenclature

N	=	total number of vertiports/nodes in the UAM network
V	=	set of all vertiports/nodes
E	=	set of all vertiport connections/edges
A	=	connectivity matrix for the vertiports/nodes
G	=	(V,E,A) UAM network expressed as a graph
K	=	set of all eVTOLs
N_K	=	number of eVTOLs
k	=	index to denote eVTOLS in K
x_{kt}	=	vertiport where eVTOL k is at time t
J_k	=	set of all journey of eVTOL $k \in K$
$Loc^k_{\text{start}}(jk)$	=	set of start vertiports for eVTOL $k \in K$ for a journey $jk \in J_k$
$Loc^k_{\text{end}}(jk)$	=	set of end vertiports for eVTOL $k \in K$ for a journey $jk \in J_k$
C	=	maximum passenger capacity for the eVTOLs
T	=	total time window for daily operation
$T^k_{\text{start}}(jk)$	=	start time of journey $jk \in J_k$ for eVTOL $k \in K$
$T^k_{\text{end}}(jk)$	=	end time of journey $jk \in J_k$ for eVTOL $k \in K$
t_{wait}	=	waiting time
δT_L	=	time allocated for landing
t	=	representation of a time instant (t $\in T$)
$U^i_{\max}$	=	eVTOL capacity of vertiport i

*Graduate Student, Department of Mechanical and Aerospace Engineering, University at Buffalo, AIAA student member
[†]Associate Professor, Department of Mechanical and Aerospace Engineering, University at Buffalo, AIAA senior member

U_t^i	=	number of eVTOLS at vertiport i at time instant t
$B_{\max}^k$	=	maximum battery capacity of eVTOL k
B_t^k	=	battery charge of eVTOL k at time instant t
$B_{\text{start}}^k(jk)$	=	battery charge for eVTOL $k \in K$ at the start of journey $jk \in J_k$
$B_{\text{end}}^k(jk)$	=	battery charge for eVTOL $k \in K$ at the end of journey $jk \in J_k$
γ	=	eVTOL battery self discharge rate
Q	=	forcasted demand model
$Q(i, j, t)$	=	forecasted demand from node i to node j at time t
P_{ijt}	=	fare charged from passengers travelling from node i to j at time t
R_{ij}	=	cost of transporting 1 passenger from vertiport i to node j
$N_k^P(jk)$	=	number of passengers transported by eVTOL k on journey $jk \in J_k$
$P_{\max}$	=	maximum power transferred between an eVTOL and the power grid
E^P	=	locational marginal pricing model for electricity
E_t^P	=	price of electricity at time instant t
g_{ijk}	=	charging/discharging rate of eVTOL k from i to j

II. Introduction

Traffic congestion has become a prominent issue in most parts of the world. This is further aggravated in larger metropolitan cities such as New York, Los Angeles, London, Mumbai etc. The average commuting time has increased from 25 minutes in the year of 2006 to 27.6 minutes in the year of 2019, which is a 10% increase as discussed in [1]. The monetary loss of this problem is about $87 billion annually [2]. Over the last few years, there has been a growing interest on **Urban Air Mobility** (UAM) based on electric vertical take-off and landing (eVTOL) for mitigating such traffic congestion. UAMs can be used for operations such as passenger transport, cargo delivery, and time-critical operations such as medical evacuation and air-ambulance services. Various private stakeholders in UAM are planning to begin their commercial services as early as 2024 [3], UAMs are projected to have a potential market size of $9.1 billion by the year 2030 [4]. In order to deploy a UAM network comprising a significant number of eVTOL aircraft operating over a region and associated vertiports (and charging stations) to support that operation, there needs to be an effective fleet scheduling framework similar to that used by commercial airlines and ride share services. Optimal fleet scheduling in this context must adapt to demand, revenue goals, aircraft constraints (such as flight range) and overall energy footprint and impact on the electricity grid. Such a scheduling application typically appears as a class of combinatorial optimization (CO) problems [5]. These CO problems can be potentially solved using a variety of classical optimization and learning-driven approaches as briefly described below.

.

A. Related Work

UAM fleet scheduling involves scheduling the journey of the eVTOLs in the UAM network at different time between the vertiports of the network in such a way to minimize the cost of operation and to maximize the revenue generated. The scheduling involves transportation of passengers between the vertiports and also for other aspects such as battery charging for the eVTOL. The UAM fleet scheduling problem can be considered as a class of multi-agent CO problems that shares characteristics with the well-known multi-Travelling Salesman Problems (mTSP) and more complex Multi-Robot Task Allocation (MRTA) problems. A majority of these problems tend to be NP-hard [6] and cannot be solved with polynomial time using traditional methods such as (Mixed) Integer (Non-)Linear Programming (ILP, MILP, MINLP etc.) [7, 8], or metaheuristics such as Ant Colony Optimization [9, 10] and genetic algorithms ([11, 12]). Even though some of these methods can generate local optimal solutions for small sized UAM fleet scheduling problems [13–15], the computational expense becomes intractable in applications where online (near real-time) decisions are required.

In recent years, a rich body of work has emerged on using learning-based techniques to model solutions or intelligent heuristics for CO problems over graphs [16–21]. A notable fraction of these methods formulates the CO problem as a Markov Decision Process (MDP) and uses Reinforcement Learning (RL) to generate policies that can yield optimal solutions across a reasonable range of problem instances [16–19, 22]. Such policies are often embodied by a trained graph neural network or GNN. The main advantages of a learning approach over classical non-learning methods include [6]: i) the ability to generalize across problem scenarios and uncertainties without tedious hand-crafting of the heuristics, ii) orders of magnitude faster run-time execution, which is critical for online task allocation and planning applications, and iii) the ability (at least in theory) to automatically learn the problem features of interest which may not be readily evident even to human experts when applications involve complex cyber-physical systems. However, the types of problems that have been typically considered in the above mentioned work (albeit with few exceptions that we will discuss later) are simple CO problems such as vanilla versions of the Travelling Salesman Problem (TSP), Vehicle Routing Problem (VRP), and Max-Cut problems [16, 19]. Up to a certain scale and specificity of planning and scheduling applications, as opposed to being able to generalize across wide range of problem classes, the complexity of the problems does not necessarily favor a learning based approach over non-learning based optimization, graph matching and local search heuristics [23, 24]. Another major limitation of most existing learning-based methods is their inability to readily apply a trained model on problem scenarios of greater complexity (e.g., multi-TSP instances with more travellers and/or locations to visit) than that encountered in the scenarios or sample episodes used for training. Retraining is usually required to currently address this issue, which can quickly become computationally prohibitive.

To address the above-stated issues, in this work we are particularly interested in exploring and advancing a class of RL methods that train models to directly encode the policy over graph space. There has been a growing interest in using sequence-to-sequence models, e.g., pointer networks and attention mechanism, to encode and learn policies for combinatorial optimization problems in graph space [16, 19]. For example, [16] implemented a framework using an encoder/decoder architecture based on attention mechanism and REINFORCE algorithm for solving a wide variety of CO problem as graphs, with the main contribution being flexible of using the approach on multiple problems with the same hyper parameters. This method has however been only applied to benchmark problems where the costs of simulating episodes is rather insignificant, unlike simulating the operation of an UAM fleet.

While adopting some of the concepts presented by Kool et al. [16], here we develop fundamental extensions to the AI architecture in order to both improve training performance and scalability, thereby making it more feasible for application to the complex and expensive (to simulate) problem of UAM scheduling. Specifically, we design a novel **encoder-decoder** policy network. Here, the **encoder** is based on Graph Capsule Convolutional Neural Networks (GCAPCN) [25], which is hypothesized to uniquely incorporate local and global structural information of the network under study (in this case the UAM network) with permutation invariance. The **decoder** is based on a Multi-head Attention mechanism (MHA) [16, 26] which fuses the encoded information and problem-specific **context** using matrix multiplication, in order to enable sequential decisions. Our overall learning architecture (also known as GCAPS-RL) consists of a policy gradient RL algorithms and the GCAPCN-MHA based policy network.

B. Main Contributions

The main contributions of this paper can thus be summarized as: **1)** Formulating the UAM fleet scheduling problem as a Markov Decision Process or MDP over graphs, with the state of the UAM network vertiport computed as embeddings of a Graph Neural Network (GNN); the states of the eVTOL aircraft in operation embedded as the *context* portion of the policy model; and a Multi-head attention (MHA) based action decoder generating the fleet scheduling actions. **2)** Exploring how a policy network comprising the above-stated encoder-decoder structure can be trained effectively through RL working in tandem with a simulation environment for UAM operations. **3)** Demonstrating this graph learning framework's ability to generalize across unseen scenarios.

Paper Outline: The next section provides a detailed description of the UAM fleet scheduling problem considered in this paper, as well as its optimization and MDP formulations. Section IV explains the proposed solution, with adequate details and equations for the state encoding (using the GNN and a feedforward network) and the MHA based action decoding by the policy network. Section V discusses the training procedure as well as the test for generalizability and performance comparison. Section VI provides concluding remarks as well as the future execution plan for generating the results for this paper.

III. Problem Description and Formulation

In this work, we consider a UAM network problem inspired by [14], involving N vertiports (also to be called "nodes" here onward), and N_K number of eVTOLs, with each eVTOL having a maximum passenger seating capacity of C. Let V and K be the set of all vertiports and eVTOLs respectively. Each vertiport $i \in V$ has a maximum number ($U^i_{\max}$) of eVTOLs it can accommodate at a time. For computing the cost of transportation, we define R_{ij} to be the cost of transporting a passengers from vertiport i to j. Unlike in [14] which considers three cases, involving **1)** scheduling and pricing, **2)** Scheduling, pricing, and frequency regulation, **3)** Frequency regulation. In this paper, we are tackling just fleet scheduling with constant pricing strategy, while planning to demonstrate the effectiveness of solving the problem on larger UAM networks than what has been studied in the literature. We consider the following assumptions for this problem:

- Every eVTOL can commute between any two vertiports
- Constant passenger pricing, which means the price of a journey for a passenger between one vertiport to the other does not change with time or other factors. Therefore P_{ijt} is a constant.
- The resistive loss of the batteries are negligible.

The aim is to maximize the profit margin by maximizing the revenue from the passenger and minimizing the operational cost and the cost of charging. This is achieved by smartly scheduling each eVTOL's journey (between vertiports to transport passengers) between the vertiports to satisfy the travel demand between each pair of vertiports. During each decision-making instant $t \in T$, an eVTOL which requires a decision will be assigned a vertiport (based on the demand, battery charge, operation cost etc.), or to wait. Below, we describe the decision making time perspectives, the passenger demand model, the battery model, and the optimization formulation of the problem.

A. Time Horizon and Time instants

We consider an operating time horizon of T (which will be considered as a normal daily operational hour). Here, a journey or trip is defined as the commute of an eVTOL between two vertiports i and j. If $i = j$, it means the eVTOL has to wait for a time $t_{\text{wait}} = 15 minutes$. Each eVTOL can take off at any time within the time horizon T. Here, we consider the schedule for a single day operation, where the time operation is from 6:00 AM (t_{start}) until 6:00PM (t_{end}), which mean the first takeoff time for any eVTOL can happen only happen from t_{start} and the last takeoff time for any eVTOL cannot be after t_{end}, during daily operations. A specific time window ($\delta T_T O$) has been allocated for the take off operation itself. Similarly a specific time window (δT_L) has been allocated for landing as well. For this study, $\delta T_{TO} = \delta T_L = 15\ minutes$. For each eVTOL k, we consider a set (J_k) of all the journeys/trips made by the eVTOL within the time horizon. Each journey $jk \in J_k$ has a start time $T^k_{\text{start}}(jk)$ and an end time $T^k_{\text{end}}(jk)$, where T^k_{start} and T^k_{end} are the set of starting time and ending time for all the journeys of eVTOL k. Let $Loc^k_{\text{start}}(jk)$ and $Loc^k_{end}(jk)$ be the start and end location corresponding to journey $jk \in J_k$. Let $T^k_{\text{journey}}(i, j)$ be the journey time of eVTOL k from vertiport i to j.

B. Passenger Fare Pricing Model

We consider a fixed pricing policy for each route depending on the operational cost and the forecast demand, similar to [27]. Therefore, the passenger fare at a time instant t between vertiport i and j, P_{ijt} will be a function of $Q(i, j, t)$ and $R_{i,j}$.

C. Demand Model

The passenger demand modeling will be used to stochastically generate the number of passenger request for travelling between different vertiports. A forecasted request (Q) has been modeled based on the data from [28]. We assume that the demand for each hour during the daily operations hours are known in prior. The demands for the trip are modeled in such a way that it resembles a subway train demand in a major city, which has subset of all the stations to be busy compared to the other stations, and generally resembles offices, work places etc. Similarly, here we consider a subset of the available vertiports $V_B \subset V$ to be high demand vertiports compared to the other vertiports. The demand between the vertiports in V_B are higher compared to vertiports in $V - V_B$. We consider two peak hours throughout the whole day from $8:00 - 9:00 AM$ (T_{peak_1}) and from $4:00 - 5:00 PM$ (T_{peak_2}). The vertiports in V_B will have both the peak hours. The journey demands from vertiports $V - V_B$ to V_B will have a peak at (T_{peak_1}), which resembles the morning rush hour for commute to workplace from home, while the journey demands from V_B to $V - V_B$ will have a peak at T_{peak_2}, which resembles the rush hour for commute from workplace to home.

$$Q(i,j,t) = \begin{cases} \mathcal{N}(100,10)\ i \in V_B, j \in V_B i \neq j,\ t = T_{\text{peak}_1}\ or\ t = T_{\text{peak}_2} \\ \mathcal{N}(100,10)\ i \in V - V_B, j \in V_B i \neq j,\ t = T_{\text{peak}_1} \\ \mathcal{N}(100,10)\ i \in V_B, j \in V - V_B i \neq j,\ t = T_{\text{peak}_2} \\ \mathcal{N}(50,10)\ i \in V_B, j \in V_B i \neq j,\ t \neq T_{\text{peak}_1}\ and\ t \neq T_{\text{peak}_2} \\ \mathcal{N}(30,5)\ \text{otherwise} \end{cases} \tag{1}$$

where $\mathcal{N}(\mu, \sigma)$, represents a normal distribution with mean μ and standard deviation σ.

D. eVTOL model

The eVTOL vehicle model considered here is the City Airbus eVTOL aircraft, with a maximum cruise speed of $74.5 mph$. This vehicle has a maximum passenger capacity of 4. The operating cost of the vehicle is about \$0.64 per mile [14].

E. Battery Model

We consider the battery model for this work is the same as that in [14], which consists of charging/discharging rate (g_{ij}, between vertiports i and j), and a self discharge rate γ. If B_t^k is the battery charge of eVTOL k at time instant t, and assuming the eVTOL travels from vertiport i to j, the battery charge after a time δT can be computed as $B_{t+1}^k = (1-\gamma)B_t^k - g_{ijk}\delta T$. We consider the maximum capacity $B_{\max}^k (k \in K)$ of the eVTOL battery to be $110 kWh$, and the maximum charging rate $P_{\max}$ to be $150 kW$. Let $B_{\text{start}}^k(jk)$ and $B_{\text{end}}^k(jk)$, be the battery charge of eVTOL k during the start and end of journey $jk \in J_k$. Here we consider that all the eVTOLs starts a new journey with full battery charge.

F. Electricity Pricing Model

The electricity pricing E^P can be modeled based on the historical market price from PJM [29]. However, since the main contribution of this work is mainly on the learning based approach on UAM fleet scheduling, we are relaxing the electricity pricing model, by assuming the price to be constant. We consider the electricity price E_p to be 20 $cents/kWh$ [30].

G. Optimization formulation:

The objective function for the optimization is to maximize the profit generated (z). The profit generated can be calculated by subtracting operational cost C^O and the cost of charging C^C from the total revenue generated R^T. The total operational cost for the time horizon T can be computed as in Eq. 2.

$$C^O = \sum_{jk \in J_k} \sum_{k \in K} N_k^P(jk) R_{ij},\ i = Loc_{\text{start}}^k(jk),\ j = Loc_{\text{end}}^k(jk) \tag{2}$$

The cost of charging C^C can be computed as:

$$C^C = \sum_{jk \in J_k} \sum_{k \in K} E_t^P g_{ijk},\ i = Loc_{\text{start}}^k(jk),\ j = Loc_{\text{end}}^k(jk) \tag{3}$$

The total revenue generated during the time horizon T can be computed as in Eq. 4

$$R^T = \sum_{jk \in J_k} \sum_{k \in K} N_k^P(jk) P_{ijt},\ i = Loc_{\text{start}}^k(jk),\ j = Loc_{\text{end}}^k(jk) \tag{4}$$

Therefore, the objective function can be formulated as:

$$\max z = R^T - C^O - C^C \tag{5}$$

Subject to:

$$T_{\text{start}}^k(jk) = 0 \ \ jk \in J_k,\ if\ t = t_{\text{start}},\ k \in K \tag{6}$$

$$\begin{aligned} T_{\text{end}}^k(jk) = T_{\text{start}}^k(jk) + \delta T_{TO} + T_{\text{journey}}^k(Loc_{\text{start}}^k(jk), Loc_{\text{start}}^k(jk)) + \delta T_L \\ if\ Loc_{\text{start}}^k(jk) \neq Loc_{\text{end}}^k(jk),\ k \in K,\ jk \in J_k \end{aligned} \tag{7}$$

$$T_{\text{end}}^k(jk) = T_{\text{start}}^k(jk) + t_{wait}\ \ if\ Loc_{\text{start}}^k(jk) = Loc_{\text{end}}^k(jk),\ k \in K,\ jk \in J_k \tag{8}$$

$$T_{\text{start}}^k(jk) = T_{\text{end}}^k(jk-1) + T_{charge}\ \ if\ Loc_{\text{start}}^k(jk-1) \neq Loc_{\text{end}}^k(jk-1),\ t > t_{\text{start}},\ k \in K,\ jk \in J_k \tag{9}$$

$$T_{\text{start}}^k(jk) = T_{\text{end}}^k(jk-1)\ \ if\ Loc_{\text{start}}^k(jk) = Loc_{\text{end}}^k(jk),\ t > t_{\text{start}},\ k \in K,\ jk \in J_k \tag{10}$$

$$N_k^P(jk) \le C,\ jk \in J_k,\ k \in K,\ t \in T \tag{11}$$

$$0 \le B_t^k \le B_{\max}^k,\ k \in K,\ t \in T \tag{12}$$

$$B^k_{\text{end}}(jk) = (1-\gamma)B^k_{\text{start}}(jk) - g_{ijk}(T^k_{\text{end}}(jk) - T^k_{\text{start}}(jk))\ k \in K,\ jk \in J_k, jk > 0 \tag{13}$$

$$B^k_{\text{start}}(jk) = B^k_{\text{max}}\ k \in K,\ jk \in J_k \tag{14}$$

$$B^k_{\text{start}}(jk) = B^k_{\text{end}}(jk-1) + T_{\text{charge}} * P_{max}\ k \in K,\ jk \in J_k \tag{15}$$

$$0 \le U^i_t \le U^i_{\text{max}},\ i \in V,\ t \in T \tag{16}$$

$$0 \le g_{ijk} \le P_{\text{max}},\ i, j \in V,\ k \in K \tag{17}$$

H. MDP Formulation

In this work, we express the fleet scheduling as a MDP, which sequentially computes the action for each eVTOL during a time instance $t \in T$. During a decision making time instance, an action will be assigned to each eVTOL (as described in the introduction of this section), based on the current state of the network. The state should have all the necessary information to assign the action. Figure 1 depicts, the how the trained policy network is being used for sequential decision making. The state, action, reward formulation, and the transition are described below.

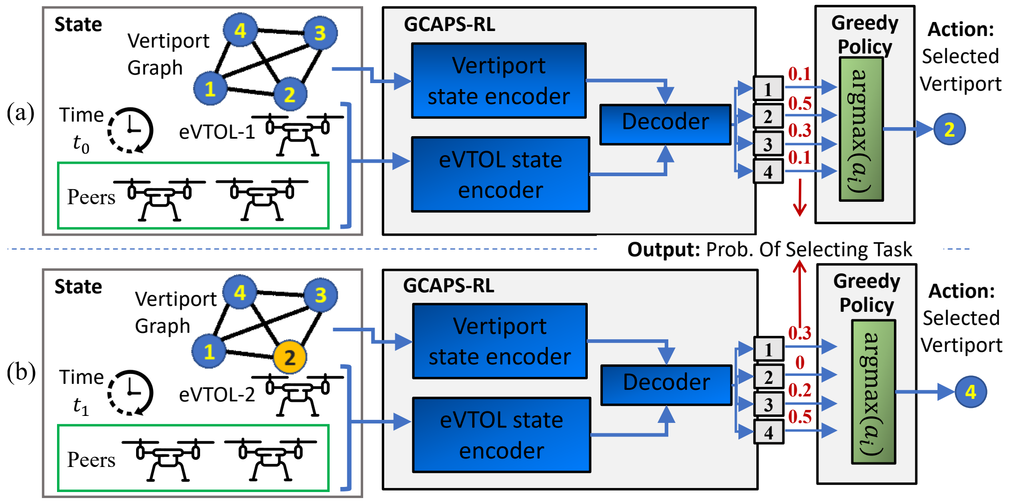

Fig. 1 Sequential decision using the policy network trained by our approach. During each decision making instance the information from the vertiport network and the eVTOLs are used to find which vertiport to visit (or to wait) for an eVTOL taking a decision.

State Space: The state information which will be used for computing the action at a time instance t consists of the information at time instance t which includes **1)** the price for electricity (E^P_t), **2)** cost of transporting a passenger ($R_{ij}, i, j \in V$), **3)** number of eVTOLs at the vertiport ($U^i_t, \forall\ i \in V$), **4)** forecasted demand ($Q(i, j, t),\ \forall\ i, j\ \in V$), **5)** charging/discharging rate of the eVTOLs ($g_{ijk}, \forall\ i, j\ \in V, k \in K$), **6)** current location of the eVTOLs ($x_{kt}, \forall\ k \in K$), and **7)** passenger fare ($P_{ijt}, \forall\ i, j\ \in V$).

Since the natural state space is large, we represent the state information as a learned feature vector of fixed length. To this end, we will represent all the information associated to the vertiports as a Graph and use a Graph Neural Network (GNN) to compute the feature vector associated with the vertiports, while the information associated with the eVTOLs will be represented as a feature vector using a simple feedforward network, which will be discussed in section IV.

Action Space: At each decision making time instance, each agent takes an action from the available action space. The action space consists of all the available vertiports. Therefore the action space will be of size N_K. During a decision making step, if an agent chooses the vertiport at which it currently is, then it waits for 15 minutes in the vertiport, until it makes a new decision.

Reward: We consider a delayed reward strategy, where the total reward at the end of the episode is the ratio of the profit (as in Eq. 5) to the maximum total profit that can be obtained ($\sum_{i\in V, j\in V, t\in T}(Q(i, j, t) \times P_{ijt})$) for that episode. Each episode corresponds to one full day operation from $6:00\ AM$ until $6:00\ PM$. Unlike [14], we do not consider a fixed time for a journey, hence the number of decision-making steps per episode is not a constant.

Transition: Since demand and electricity pricing can vary from that of the forecasted values, the transition of the states is considered to be stochastic.

The transition is an event-based trigger. An event is defined as the condition that an eVTOL is ready for takeoff. As environmental uncertainties and communication issues (thus partial observation) are not considered in this paper, only deterministic state transitions are allowed. The size of the different variables of the state space and the action space are shown in table 1[*].

Table 1 State space and action space variables of the MDP formulation

	Parameter	Size
State space	1) Price of electricity	1
	2) Transportation cost	$N \times N$
	3) Number of eVTOLs at the vertiport	N
	4) Forecasted demand	$N \times N \times 13$
	5) Charge/discharge rate for the eVTOLs	N_K
	6) Current location of the eVTOLs	N_K
	7) Passenger fare	$N \times N$
Action space	Vertiport to be visited	1

IV. Proposed Learning-based Solution Approach

In this work, we propose a policy gradient RL based method to train a policy network, which will be used to assign actions to the eVTOLs during each decision making time instance . The policy network takes in the state information during each decision making time instance and output an action for each eVTOL. The policy network consists of GNN based encoder, a context module, and a Multi-head attention (MHA) based decoder. The state information is being encoded as learnable fixed length feature vectors by the encoder and the context module, and will be used by the decoder to compute the action sequentially. Further information regarding the proposed state encoding and the action decoding is being discussed in the following section.

A. State Encoding

The state information consists of the information of UAM vertiports as well as the information regarding the eVTOLs. The vertiports information is formulated as a graph as described below in section IV.A.1. The feature vector for the eVTOL state information is discussed in section IV.A.2.

1. Vertiport State Encoding

Graph Formulation for UAM Network: The UAM network can be expressed a graph $G = (V, E, A)$, where V represents the set of nodes or the set of vertiports in this case, E represents the set of edges between the nodes, A represents the adjacency matrix of the nodes. Since there are no restrictions on the eVTOLs for commuting between any two vertiports, we assume G to be undirected and fully connected. Each node $i \in V$ has its time varying node properties δ_i^t. Here the properties of each node $i \in V$ at time instance t are: **1)** number of eVTOLs at i (U_t^i), **2)** eVTOL capacity at i (U_{max}^i), **3)** the predicted total number of passengers who wants to depart from i (d_i^t), **4)** the predicted total number of passengers who want to reach i (r_i^t), **5)** passenger fare from node i to all the other vertiports ($P_{i1t}, \ldots, P_{iNt}$), **6)** cost of transporting a passenger from node i to all the other vertiports at time instance t ($R_{i1}, \ldots, R_{iN}$). Therefore, $\delta_i^t = [U_t^i, U_{max}^i, d_i^t, r_i^t, P_{i1t}, \ldots P_{iNt}, R_{i1}, \ldots, R_{iN}]$. Therefore the size of δ_i^t is $\mathbb{R}^{2N+4}$.

The main purpose of the encoder, is to represent useful information related to a node/vertiport as a learnable continuous vector or tensor, which can then be used by the learning algorithm. General solutions to combinatorial optimization problems such as CVRP, MTSP and MRTA should be permutation invariant, which means that the order by which each node is numbered (or indexed) should not affect the optimal solution. Hence the node encoding must also be permutation invariant. In this work, we are exploring how a Graph Capsule Convolutional Neural Network (GCAPCN) can be implemented for learning local and global structures with the node properties, with permutation invariant node embedding. GCAPCN is a class of Graph Neural Networks (GNN), introduced in [25] to address the

[*] N is the number of vertiports, and N_K is the number of eVTOLs

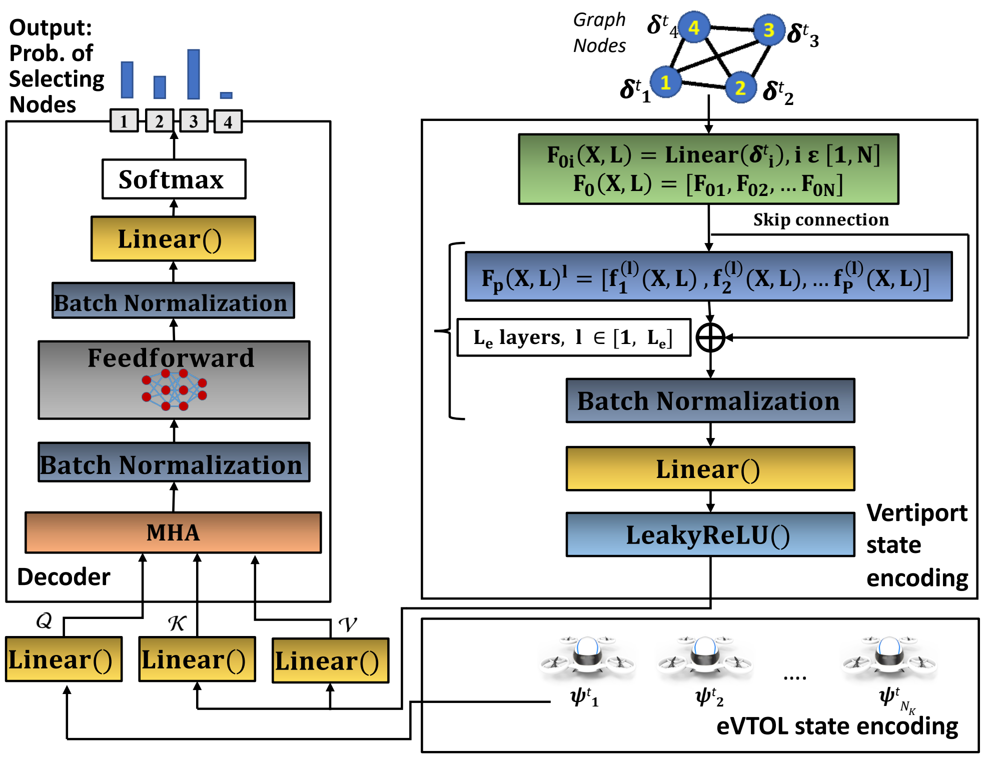

Fig. 2 Policy network architecture, which includes the vertiport state encoder, the eVTOLs state encoder, and the action decoder

drawbacks (e.g., permutation invariance) of Graph Convolutional Neural Networks (GCN), and to enable the encoding of global information based on **capsule networks** presented in [31]. The advantage of GCAPCN lies in capturing more local and global structural information from the graph under study, compared to conventional aggregation operations used in GNN such as summation or standard convolution operations.

Graph Capsule Convolutional Neural Networks: Let $X \in \mathbb{R}^{N\times|\delta^t_i|}$, be the node feature matrix, where $|\delta^t_i|$ is the input dimension for each node i. The standard graph Laplacian is defined as $L = D - A \in \mathbb{R}^{N\times N}$, where D is the degree matrix and A is the adjacency matrix of the graph. A capsule vector is computed using a Graph Capsule function based on different order of statistical moments, as shown in the equations later in this section.

We first compute a feature vector F_{0i} for each node by linear transformation of the node properties δ^t_i, as $F_{0i} = \delta^t_i.W_0 + b_0$ for all $i \in [1, N]$, where $W_0 \ \varepsilon \ \mathbb{R}^{h_0\times|\delta_i|}$, $b_0 \ \varepsilon \ \mathbb{R}^{h_0\times 1}$, and h_0 is the length of the feature vector.

Each feature vector $F_{0i}, i \in [1, N]$ is then passed through a series of Graph capsule layers, where the output from the previous layers is used to compute a matrix $f_p^{(l)}(X, L)$ using a graph convolutional filter of polynomial form as given by:

$$f_p^{(l)}(X,L) = \sigma\left(\sum_{a=0}^{\alpha} L^a (F_{(l-1)}(X,L)^{\circ p}) W_{pa}^{(l)}\right) \tag{18}$$

Here L is the graph Laplacian, p is the order of the statistical moment, α is the degree of the convolutional filter, $F_{(l-1)}(X, L)$ is the output from layer $l - 1$, $F_{(l-1)}(X, L)^{\circ p}$ represents p times element-wise multiplication of $F_{(l-1)}(X, L)$. Here, $F_{(l-1)}(X, L) \in \mathbb{R}^{N\times h_{l-1}p}$, $W_{pa}^{(l)} \in \mathbb{R}^{h_{l-1}p\times h_l}$. The variable $f_p^{(l)}(X, L) \in \mathbb{R}^{N\times h_l}$ is a matrix where each row is an intermediate feature vector for each node $i \in [1, N]$, infusing nodal information from $L_e \times \alpha$ hop neighbors, for a value of p. The output of layer l is obtained by concatenating all $f_p^{(l)}(X, L)$, as given by:

$$F_l(X,L) = [f_1^{(l)}(X,L), f_2^{(l)}(X,L), \ldots f_P^{(l)}(X,L)] \tag{19}$$

Here P is the highest order of statistical moment, and h_l is the node embedding length of layer l. We consider all the values of h_l (where $l \in [0, L_e]$) to be the same for this paper. Equations 18 and 19 were computed for L_e layers, where each layer uses the output from the previous layer ($F_{l-1}(X, L)$). Adding more layers helps in learning the global structure, however, this can affect the performance by increasing the number of learnable parameters (compared to the size of the problem), leading to over-fitting. The output from the final layer is then passed through a feed-forward layer so that the final feature vector has the right dimension (h_l) to be fed into the decoder as shown in Fig. 2

2. *eVTOL State Information Encoding*

The state information corresponding to the eVTOLS at time instance t consists of **1)** the current electricity price (E_t^P), **2)** the current location of the eVTOL ($x_{kt}, k \in K$), **3)** the current battery charge of the eVTOL ($B_t^k, k \in K$), **4)** the discharge rate of the eVTOL from its current vertiport to all the other vertiports ($g_{x_{kt}1k}, \ldots, g_{x_{kt}Nk}, k \in K$). The eVTOL information state properties can be represented as $\psi_k^t = [E_t^P, x_{kt}, B_t^k, g_{x_{kt}1k} \ldots, g_{x_{kt}Nk}]$, where $k \in K$.

B. Action Decoding

The main objective of the decoder is to use the information from the vertiport graph encoding and the eVTOL information encoding as context or query, and thereof choose the best vertiport by calculating the probability value of getting selected for each (vertiport) node. In this case, the first step is to feed the embedding for each node (from the encoder) as **key-values** ($\mathcal{K}$, $\mathcal{V}$). The key $\mathcal{K}$ and value $\mathcal{V}$ for each node is computed by two separate linear transformations of the node embedding obtained from the encoder. The next step is to compute a vector by a linear transformation of also known as the **context** Q is computed a linear transformation of ψ_k.

Now the attention mechanism can be described as mapping the query (Q) to a set of key-value ($\mathcal{K}, \mathcal{V}$) pairs. The inputs, which are the query (Q) is a vector, while $\mathcal{K}$ and $\mathcal{V}$ are matrices of size $d_{embed} \times N$ (since there are N nodes). The output is a weighted sum of the values $\mathcal{V}$, with the weight vector computed using the compatibility function expressed as:

$$\text{Attention}(\mathcal{K}, \mathcal{V}, Q) = \text{softmax}(Q^T \mathcal{K} / \sqrt{d_{embed}}) \mathcal{V}^T \tag{20}$$

Here h_l is the dimension of the key of any node i ($k_i \in \mathcal{K}$). In this work, we implement a multi-head attention (MHA) layer in order to determine the compatibility of Q with $\mathcal{K}$ and $\mathcal{V}$. The MHA implemented in this work is similar to the decoder implemented in [16] and [26]. As shown in [26] the MHA layer can be defined as:

$$\text{MHA}(\mathcal{K}, \mathcal{V}, Q) = \text{Linear}(\text{Concat}(\text{head}_1 \ldots \text{head}_{h_e})) \tag{21}$$

Here $\text{head}_i = \text{Attention}(\mathcal{K}, \mathcal{V}, Q)$ and h_e (taken as 8 here) is the number of heads. The feed-forward layer is implemented to further process the mapping that results from the MHA layer, and transform it to a dimension that is coherent with the number of nodes in the task-graph (N). The interjecting batch normalization layers serve to bound values of a specific batch using the mean and variance of the batch. The final `softmax` layer outputs the probability values for all the nodes. Here, the next task to be done is then chosen based on a greedy approach, which means that the node with the highest probability will be chosen.

C. Simulation Environment

The simulation environment has been implemented in *Python*, following the **Open AI Gym environment** interface. The environment was created following the MDP formulation in section III.H and the constraints specified in section III are explicitly enforced in the simulation environment.

V. Experimental Evaluation

A. Training details

Scenario Description: We considered a hypothetical location of 50 x 50 sq. miles area, and consisting of 8 vertiport locations. Out of the 8 vertiports, 3 of them are considered to be high traffic vertiports, which means the trips demand between these 3 vertiports are high compared to the other trips. For every training scenario, we consider the veritport location to be the same. The hourly demand values $Q(i, j, t)$ changes for each episode as described in section III.C. The initial locations of the eVTOLs during each episode will be different. The *"Python"* 3.7 and the 64-bit distribution of *"Anaconda 2020.02"* are used to implement the MRTA approaches. The environment, training algorithm, and the evaluation of the trained model, are all implemented in *Pytorch-1.5*. The training, based on Pytorch, is deployed on two GPUs (NVIDIA Tesla V100) with 16GB RAM.

Training Algorithm In order to train the policy network, we implemented a policy gradient RL method, Proximal Policy Optimization [32]. The policy gradient RL algorithm is implemented using the off-the-shelf algorithm available from **stable-baselines3** [†]. The relevant settings used for the training can be found in table 2, and the settings for the policy network during training can be found in table 3.

[†]https://stable-baselines3.readthedocs.io/en/master/guide/algos.html

Table 2 Settings for model training

Details	Values
Algorithm	*PPO*
Total steps	2,000,000
Rollout buffer size	20000
Batch size	10000
Optimizer	*Adam*
Learning step size	0.000001
Entropy coefficient	0.01
Value function coefficient	0.5
Epochs	100

Table 3 Settings for the policy network

Details	Values
Embedding length (h)	256
Highest order of statistical moment (P)	2
Degree of convolutional filter (K)	2
Number of layers (L_e)	1
Number of attention heads in the decoder (h_e)	8

B. Performance testing

We design and execute a set of numerical experiments, to investigate the performance of our proposed learning-based algorithm over graph space (*GCAPS-RL*) and compare it with 1) another learning based method with a Multi-layered perceptron as the policy network (MLP-RL); 2) a myopic baseline called Feasibility-preserving Random-Walk (Feas-RND) that takes randomized but feasible actions (avoiding conflicts and satisfying other problem constraints). The Feas-RND method provides a baseline that GCAPS-RL should clearly surpass in performance (total reward), in order to demonstrate that meaningful = policies are being learnt as opposed to simply mapping random feasible actions.

1. Generalizabilty

Generalizability refers to the ability of the learned model to display similar performance on unseen scenarios compared to the scenarios on which it was trained for. In order to compare the convergence of the proposed GCAPS-RL method with that of the MLP-RL approach, we run both methods with similar settings and plot their learning curve (convergence history), as shown in Fig. 3. As seen from this figure, the GCAP-RL converged to a larger total reward (0.042). compared to MLP-RL (0.034). It can be seen that for both the methods, during the initial phase of training (<500000 steps), the average total reward fluctuates, and then later on the fluctuation decreases significantly as the training progresses. This is because we set an entropy coefficient of 0.01, which forces both the models to explore.

Test cases scenarios: We generated 100 different testing scenarios drawn from the same distribution as that for training (as explained in section III), and implemented the GCAPS-RL and the two baseline methods for performance comparison.

Figure 4 shows the comparison of the total reward per scenario for all the three methods, as boxplots. GCAPS-RL clearly demonstrates superior performance compared to both the baseline methods. In order to test for the significance of this performance superiority, we performed a statistical pairwise **t-test**. Here, the null hypothesis is that the difference between the values of the two sets has a mean equal to 0. Considering a 5% significance level. The **p-value** from the T-test for both the tests (GCAPS-RL vs Feas-RND, and GCAPS-RL vs MLP-RL) was found to be less than 0.05, which indicates the rejection of the null hypothesis – GCAPs-RL's performance on the test cases is thus significantly better

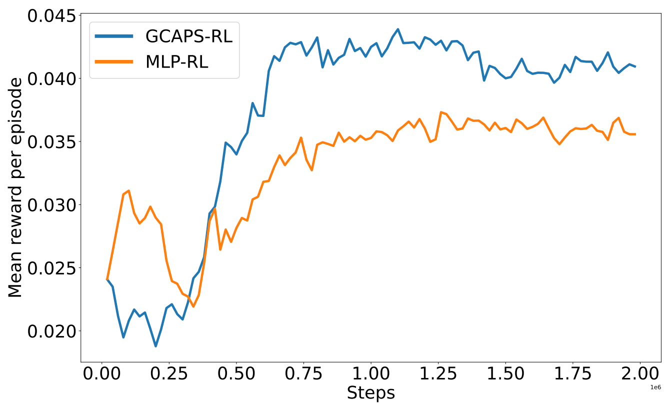

Fig. 3 Convergence plot for GCAPS-RL and MLP-RL over 2 million steps of training.

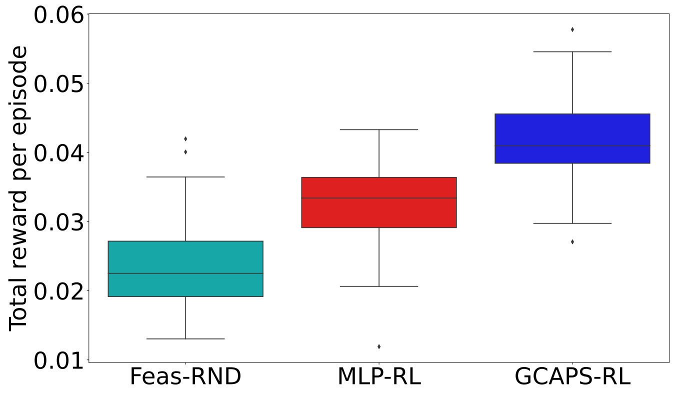

Fig. 4 Comparison of GCAPS-RL with the two baselines on average episodic reward.

than of the baselines.

Figure 5 shows the comparison of the total profit generated per day for all the three methods. The GCAPS-RL was able to generate an average daily profit of $28747 with a standard deviation of $2457, while the average daily profit for MLP-RL was $22799 with standard deviation of $3690, and for Feas-RND it was $15874 with a standard deviation of $2070. On an average, GCAPS-RL generated 26% more profit compared to MLP-RL, and 81% more profit compared to Feas-RND. The average total number of trips (excluding waiting) was found to be 350, 365, and 337, for GCAPS-RL, MLP-RL, and Feas-RND, respectively. Even though GCAPS-RL has lesser number of trips made compared MLP-RL, GCAPS-RL demonstrated superior performance. This is because GCAPS-RL was able to learn policies such that a larger number of eVTOLS are able to operate on routes with peak hours. The main reason for the superior performance of GCAPS-RL can be credited to the policy network. By formulating the vertiport network as a graph as described in section III.H, the GCAPCN has the ability to capture both the graph nodal properties as well ass the higher dimensional structural information which are beneficial for decision-making. A simple network such as MLP is not able to explicitly capture this information, and the only way to include this information is by the implicitly making the MLP network to learn. In order to learn this non-Euclidean data structural information, the MLP requires more learnable weights in the form of hidden layers, and as a result, this can lead to slower learning. The comparison with Feas-RND shows how the learning method is markedly better than random **feasible** myopic decisions, thereby indicating that meaningful policies have been learnt here, as opposed to producing random feasible solution by virtue of the masked policy network design.

VI. Conclusion

In this paper, we proposed a RL based architecture with a GNN called GCAPS-RL, to learn policies for UAM fleet scheduling problems. This new architecture incorporates an encoder based on capsule networks for vertiport state encoding, a linear encoder for eVTOLs state encoding, and a decoder based on the attention mechanism. To learn the

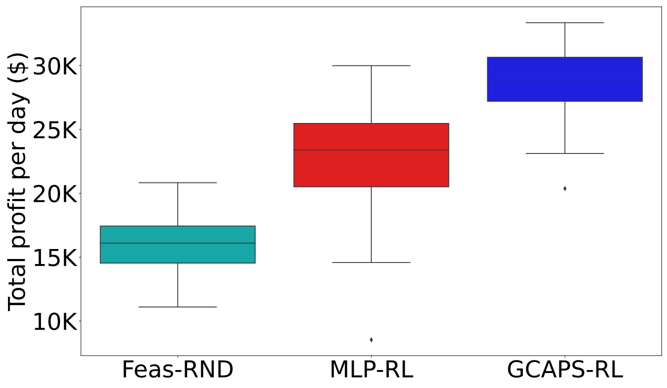

Fig. 5 Comparison of GCAPS-RL with the two baselines on average daily profit generated

features of both the encoders and decoder, the problem has been posed as an RL problem and solved using the policy gradient algorithm, Proximal Policy Optimization (PPO). In addition, the proposed architecture is found to provide effective policies over unseen scenarios. The new GCAPS-RL architecture demonstrated better performance compared to two other baselines, which include both a learning based method (MLP-RL) and a non-learning based methods Feasibility preserving Random walk (Feas-RND). The GCAPS-RL architecture with its capsule based node embedding showed that learning local and global structural information of the task graph results in better generalizability over unseen test cases, as observed from comparing its performance with MLP-RL (that uses a different node embedding).

Future directions: Firstly, to enable transition of our methods to application, in the future we should consider dynamic hourly demand, dynamic electricity pricing, environment uncertainties, and partially observable state spaces within the GCAPS-RL architecture. Another direction involves enabling the trained model to scale to larger scenarios which involves larger number of vertiports and eVTOLs, than the for which a model is trained.

Acknowledgments

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

The authors of this paper would also like to thank Prajit Krisshnakumar (University at Buffalo), and Wenyuan Li (University at Buffalo) for their valuable contribution to this work.

References

[1] "Travel Time to Work in the United States: 2019," , 2021. URL `https://www.census.gov/library/publications/2021/acs/acs-47.html`.

[2] Reed, T., "Inrix global traffic scorecard," 2019.

[3] III, W. B., "Evtol investments will continue billion dollar trend in 2021," , ???? URL `http://interactive.aviationtoday.com/avionicsmagazine/february-march-2021/evtol-investments-will-continue-billion-dollar-trend-in-2021/`.

[4] Markets, and Markets, "Urban Air Mobility Market Size Global Forecast to 2030," , 2021. URL `https://www.marketsandmarkets.com/Market-Reports/urban-air-mobility-market-251142860.html`.

[5] Hwang, S. I., and Cheng, S. T., "Combinatorial Optimization in Real-Time Scheduling: Theory and Algorithms," *Journal of Combinatorial Optimization*, 2001. https://doi.org/10.1023/A:1011449311477.

[6] Peng, Y., Choi, B., and Xu, J., "Graph Learning for Combinatorial Optimization: A Survey of State-of-the-Art," *Data Science and Engineering*, 2021, pp. 1–23.

[7] Miller, C. E., Tucker, A. W., and Zemlin, R. A., "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, Vol. 7, No. 4, 1960, pp. 326–329.

[8] Kamra, N., and Ayanian, N., “A mixed integer programming model for timed deliveries in multirobot systems,” *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, IEEE, 2015, pp. 612–617.

[9] Rizzoli, A. E., Montemanni, R., Lucibello, E., and Gambardella, L. M., “Ant colony optimization for real-world vehicle routing problems,” *Swarm Intelligence*, Vol. 1, No. 2, 2007, pp. 135–151. https://doi.org/10.1007/s11721-007-0005-x, URL https://doi.org/10.1007/s11721-007-0005-x.

[10] Wang, X., Choi, T.-M., Liu, H., and Yue, X., “Novel Ant Colony Optimization Methods for Simplifying Solution Construction in Vehicle Routing Problems,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 11, 2016, pp. 3132–3141. https://doi.org/10.1109/TITS.2016.2542264.

[11] Zhang, T., Gruver, W., and Smith, M. H., “Team scheduling by genetic search,” *Intelligent Processing and Manufacturing of Materials, 1999. IPMM’99. Proceedings of the Second International Conference on*, Vol. 2, IEEE, 1999, pp. 839–844.

[12] Mühlenbein, H., “Parallel genetic algorithms, population genetics and combinatorial optimization,” *Parallelism, Learning, Evolution*, edited by J. D. Becker, I. Eisele, and F. W. Mündemann, Springer Berlin Heidelberg, Berlin, Heidelberg, 1991, pp. 398–406.

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

[14] Shihab, S. A. M., Wei, P., Shi, J., and Yu, N., “Optimal eVTOL Fleet Dispatch for Urban Air Mobility and Power Grid Services,” *AIAA AVIATION 2020 FORUM*, 2020.

[15] Kim, S. H., “Receding Horizon Scheduling of On-Demand Urban Air Mobility With Heterogeneous Fleet,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 4, 2020, pp. 2751–2761. https://doi.org/10.1109/TAES.2019.2953417.

[16] Kool, W., Van Hoof, H., and Welling, M., “Attention, learn to solve routing problems!” *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[17] Barrett, T. D., Clements, W. R., Foerster, J. N., and Lvovsky, A. I., “Exploratory combinatorial optimization with reinforcement learning,” *arXiv preprint arXiv:1909.04063*, 2019.

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

[19] Kaempfer, Y., and Wolf, L., “Learning the Multiple Traveling Salesmen Problem with Permutation Invariant Pooling Networks,” *ArXiv*, Vol. abs/1803.09621, 2018.

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

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

[22] Jacob, R. A., Paul, S., Li, W., Chowdhury, S., Gel, Y. R., and Zhang, J., “Reconfiguring Unbalanced Distribution Networks using Reinforcement Learning over Graphs,” *2022 IEEE Texas Power and Energy Conference (TPEC)*, 2022, pp. 1–6. https://doi.org/10.1109/TPEC54980.2022.9750805.

[23] Jose, K., and Pratihar, D. K., “Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods,” *Robotics and Autonomous Systems*, Vol. 80, 2016, pp. 34–42.

[24] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., and Van Oudheusden, D., “Iterated local search for the team orienteering problem with time windows,” *Computers & Operations Research*, Vol. 36, No. 12, 2009, pp. 3281–3290. https://doi.org/https://doi.org/10.1016/j.cor.2009.03.008, URL https://www.sciencedirect.com/science/article/pii/S030505480900080X, new developments on hub location.

[25] Verma, S., and Zhang, Z. L., “Graph capsule convolutional neural networks,” , 2018.

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I., “Attention Is All You Need,” *CoRR*, Vol. abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[27] “How uber’s dynamic pricing model works,” , Nov 2021. URL https://www.uber.com/en-GB/blog/uber-dynamic-pricing/.

[28] “Our Nation’s Highways 2010,” , No. FHWA-PL-10-023, 2010. URL https://rosap.ntl.bts.gov/view/dot/904.

[29] “Data miner,” , ????. URL https://www.pjm.com/markets-and-operations/etools/data-miner-2.aspx.

[30] “See electric rates available to your home/business (updated today):,” , ????. URL https://www.electricchoice.com/electricity-prices-by-state/.

[31] Hinton, G. E., Krizhevsky, A., and Wang, S. D., “Transforming Auto-Encoders,” *Artificial Neural Networks and Machine Learning – ICANN 2011*, edited by T. Honkela, W. Duch, M. Girolami, and S. Kaski, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 44–51.

[32] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., “Proximal Policy Optimization Algorithms,” , 2017.