Deep Reinforcement Learning for UAV Routing in The Presence of Multiple Charging Stations

Mingfeng Fan, Yaoxin Wu, Tianjun Liao, Zhiguang Cao, Hongliang Guo, Guillaume Sartoretti, Guohua Wu

***Abstract*—Deploying Unmanned Aerial Vehicles (UAVs) for traffic monitoring has been a hotspot given their flexibility and broader view. However, a UAV is usually constrained by battery capacity due to limited payload. On the other hand, the development of wireless charging technology has allowed UAVs to replenish energy from charging stations. In this paper, we study a UAV routing problem in the presence of multiple charging stations (URPMCS) with the objective of minimizing the total distance traveled by the UAV during traffic monitoring. We present a deep reinforcement learning based method, where a multi-head heterogeneous attention mechanism is designed to facilitate learning a policy that automatically and sequentially constructs the route, while taking the energy consumption into account. In our method, two types of attentions are leveraged to learn the relations between monitoring targets and charging station nodes, adopting an encoder-decoder-like policy network. Moreover, we also employ a curriculum learning strategy to enhance generalization to different numbers of charging stations. Computational results show that our method outperforms conventional algorithms with higher solution quality (except for exact methods such as Gurobi) and shorter runtime in general, and also exhibits strong generalized performance on problem instances with different distributions and sizes.**

***Index Terms*—Deep reinforcement learning, UAV routing, combinatorial optimization problems, heuristics.**

I. Introduction

THE rapid urbanization and the surge in car ownership have brought around severe traffic congestion and safety issues, especially in large cities [1]. A commonly used method to mitigate those issues is monitoring the real-time traffic on roads, based on which some effective preventive measures could be taken in advance. Traditional technologies for traffic monitoring rely on sensors that are deployed in the road network, such as inductive loop detectors and video cameras, to track traffic flow. However, these devices are usually placed at fixed locations and thus are less effective in capturing information like the global view of traffic flows over space, vehicle trajectories, and the layout of road networks [2]. On the other hand, with the fast advances in unmanned aerial vehicles (UAVs) in both academia and industry [3]–[6], such as navigation technology [7], [8], UAVs have been widely exploited for traffic monitoring in recent years. With onboard cameras, UAVs are able to collect traffic information (data) and send it to a cloud-based server in real time. Compared with traditional technologies in the context of traffic data collection, UAVs enjoy a number of advantages, e.g., they are able to provide a broader view of traffic flows while also allowing to freely emphasize on local areas of interest; they can more easily handle obstacles, and relatively low-cost [9].

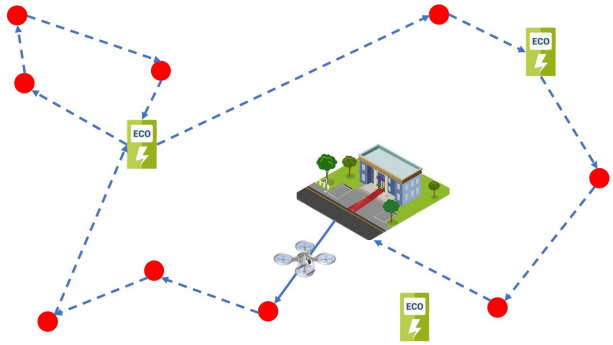


Fig. 1: Example path allowing a UAV to cover all monitoring targets while visiting several stations for recharging along the way.

Given a series of targets (nodes) to be monitored, it is critical to optimize the route of a UAV so that the information about the respective targets could be returned as fast as possible. Assuming a roughly constant speed, it is expected that the UAV covers the targets by visiting them in an order with the shortest distance. On the other hand, UAVs are often limited by the battery capacity [10], and it is impossible for a UAV to monitor a relatively large area without recharging. However, a number of charging stations spread across a city are often available to supply power to UAVs, such as in the new electric grid paradigms like Smart Grids (SG) and intermittent Renewable Energy Resources (RER), which allow the UAV to be recharged directly in independently scattered charging stations [11]. In light of those practical conditions, this paper aims to study the UAV routing problem in the presence of multiple charging stations (URPMCS) for a traffic monitoring task. Specifically, a UAV starts from the depot to collect information of a group of targets (nodes) by visiting

This work was supported by the National Natural Science Foundation of China (Grant No. 62073341), and Fundamental Research Funds for the Central Universities of Central South University (Grant No. 2022ZZTS0191) *(Corresponding author: Guohua Wu)*.

Mingfeng Fan and Guohua Wu are with School of Traffic and Transportation Engineering, Central South University, China (E-mails: mingfan@csu.edu.cn, guohuawu@csu.edu.cn).

Yaoxin Wu is with School of Computer Science and Engineering, Nanyang Technological University, Singapore (E-mail: wuyaoxin@ntu.edu.sg).

Tianjun Liao is with Academy of Military Sciences, China (E-mail: tianjunliao_ams@hotmail.com).

Zhiguang Cao and Hongliang Guo are with the Institute for Infocomm Research (I2R), A*STAR, Singapore (E-mails: zhiguangcao@outlook.com, guo_hongliang@i2r.a-star.edu.sg).

Guillaume Sartoretti is with Department of Mechanical Engineering, National University of Singapore, Singapore (E-mail: guillaume.sartoretti@nus.edu.sg).

them sequentially and returns to the depot after all targets are covered. To complete the monitoring task, the UAV usually needs to reach a charging station to replenish its battery whenever it is going to run out of power and then restarts to visit the remaining targets. The solution (route) to an example URPMCS instance is depicted in Fig. 1, where the UAV monitors (or visits) each target exactly once during the travel, while each charging station is allowed to be visited more than once or never. In this paper, we consider minimizing the total travel distance of the UAV as the objective for solving URPMCS while respecting the power constraints.

In general, URPMCS can be regarded as a variant of the classic traveling salesman problem (TSP), and thus it inherits the NP-hard characteristic of most combinatorial optimization problems. Additionally, URPMCS is also related to the green VRPs [12], since it aims to optimize routes for fuel-less vehicles. Despite its importance, less attention has been paid to URPMCS compared to other routing problems. On the other hand, conventional heuristics, e.g., simulated annealing (SA) and genetic algorithm (GA), are often used to solve NP-hard routing problems, delivering fairly good solutions in reasonable time. However, their performance highly rely on domain-specific engineering. In contrast, learning-based heuristics train neural networks to automatically construct or improve routes in a data-driven fashion. Among the paradigms, deep reinforcement learning (DRL) based methods only need the cost of any solution as the reward signal to update its policy network, in contrast to the supervised learning methods which require vast and expensive labels that are achieved by solving the routing problems in prior [13]–[15]. While DRL can generally produce high-quality solutions, most existing works focus on problems like TSP [16] and capacitated vehicle routing problem (CVRP). They will become less effective in solving URPMCS, as the targets and charging stations are two completely different types of nodes, especially given that a charging station is able to expand the travel range of a UAV and can be visited more than once or never.

Motivated by the above issues, we propose a heterogeneous attention based DRL (HADRL) method to automatically learn a construction heuristic for solving URPMCS. The problem is first modeled as a Markov decision process (MDP), and then we design the policy network in conjunction with a multi-head heterogeneous attention to learn the advanced representation of the relation between targets and charging stations. Finally, we train the network with reinforcement learning (RL) and boost its generalization against different numbers of charging stations with a curriculum learning strategy. The main contributions are summarized as follows:

1) We formulate both mathematical programming and MDP for URPMCS.
2) We exploit heterogeneous attention in policy network for relation embedding between targets and charging stations to obtain more effective feature extraction from input, which can promote the performance in route generation for URPMCS.
3) We leverage a curriculum learning strategy in the training algorithm to improve the performance and generalization to different numbers of charging stations.
4) We conduct extensive experiments and the results show that our method outperforms relevant conventional and learning-based approaches, and generalizes well across different problem sizes and distributions. Furthermore, we extend our method in multi-UAV cases (see Section V-F).

The remainder of this paper is organized as follows: Section II reviews related works; Section III presents the formulation of the URPMCS as a mathematical programming and MDP; Section IV details the proposed method and policy network; Section V presents and analyzes our experimental results; Section VI concludes this paper.

II. Related Work

In this section, we review the related works on UAV routing problem for traffic monitoring and existing DRL methods for similar vehicle routing problems, respectively.

A. UAV Routing for Traffic Monitoring

UAV routing plays a key role in the deployment of UAVs for traffic monitoring [17], where approaches for classic vehicle routing problems (VRPs), e.g., TSP and CVRP, are usually used or adapted. Among them, Shetty et al. [18] decomposed the strategic routing for UAVs into two phases, i.e., target assignment and TSP based route planning for respective UAVs. Casbeer and Holsapple [19] converted the UAV assignment into a VRP with precedence constraints and solved it with a column generation algorithm. Karaman and Frazzoli [20] transformed the multi-UAV mission planning as a VRP with temporal logic specifications. Similarly, Thibbotuwawa et al. [21] tackled UAV mission planning as a CVRP considering battery and payload weight. In addition, some other works leveraged approaches for dynamic VRPs to handle UAV routing in uncertain environments [22], [23]. Kinney et al. [24] developed an operational UAV routing system based on a quick-running routing heuristic. Guerrero and Bestaoui [25] adopted a Zermelo-TSP method to compute the optimal route for a UAV considering the influence of the wind. Savuran and Karakaya [26] studied the routing problem for a UAV on a moving carrier, and modeled it as a CVRP with a mobile depot. Different from the above ones which optimized a single objective, some other works focused on multi-objective optimization. For example, Wen et al. [27] minimized the total travel time and the fleet size of homogeneous UAVs for an extension to the CVRP, which contains two factors of distance and weight. Lamont et al. [28] proposed a multi-objective evolutionary algorithm to minimize both the cost and risk for a UAV swarm. Guerriero et al. [29] studied a UAV routing problem with time windows, whose objective is to maximize customer satisfaction while minimizing the number of used UAVs simultaneously.

Despite those diverse works, only a few of them considered *en route* refueling (or recharging) for UAVs. Sunder and Rathinam [30] studied a routing problem for UAV of fixed wing where the heading angle is constrained in the presence of several depots for refueling. Li et al. [1] investigated the UAV scheduling problem with uncertain demands for traffic

TABLE I: Comparative summary of related literature on UAV routing problems

	CS setup	UAV setup	Method
Klein et al. [22] Guerrero and Bestaoui [25] Savuran and Karakaya [26]	No CS	Single UAV	Expert knowledge
Shetty et al. [18] Casbeer and Holsapple [19] Karaman and Frazzoli [20] O'Rourke et al. [23] Kinney et al. [24] Wen et al. [27] Lamont et al. [28] Guerriero et al. [29]	No CS	Multi-UAV	Expert knowledge
Li et al. [1] Thibbotuwawa et al. [21]	Single CS (The depot)	Multi-UAV	Expert knowledge
Sundar and Rathinam [30]	Multi-CS	Single UAV	Expert knowledge
Coelho et al. [31]	Multi-CS	Multi-UAV	Expert knowledge
Our work	Multi-CS	Single UAV, Multi-UAV	Data driven

The CS is the abbreviation of charging station.

monitoring, where UAVs are only charged at the depot. Coelho et al. [31] introduced a multi-objective green UAV routing problem that allows UAVs to be recharged at charging stations.

Table I further compares the our work and the studies above. Although we have a similar problem setup as Coelho et al. [31], their main contribution lies in the mathematical programming model, while ours focuses on devising an effective routing algorithm. Furthermore, most existing methods for UAV routing are developed based on classic heuristics, which are labor-intensive for algorithmic design. They may suffer from inferior performance and inefficient computation, especially in large-scale problems.

B. Learning for Vehicle Routing

Applying DRL to solve vehicle routing problems has made a great success in recent years. Most existing deep models follow an encoder-decoder structure. The encoder learns advanced node representations from raw features, while the decoder infers solutions according to the node and graph embeddings from the encoder. There are two primary paradigms for decoding to solve the routing problems, which result in learning based *construction* or *improvement* approaches [32].

In construction approaches, the decoder picks a node at each decoding step to form a complete solution (i.e., a route). Vinyals et al. [33] designed the Pointer Network (PN) by equipping a recurrent neural network (RNN) with an attention mechanism [34], and trained it to predict the optimal tour for TSP in a supervised fashion. However, acquiring the optimal solutions to TSP instances is rather time-consuming, especially for large-scale problems. Bello et al. [35] proposed a DRL based method to train the PN, which outperforms the original one on TSP with 100 nodes. Nazari et al. [36] then discovered that encoding the permutation of the input nodes is illogical, so they replaced the RNN in PN with element-wise projections and applied it to solve CVRP. On top of [36], Kool et al. [37] exploited a Transformer-style model to solve a series of routing problems, which exhibited superior performance to several classic (meta-)heuristic methods. Subsequently, considerable methods are adapted from it and applied to solve a variety of VRPs [38]–[42].

In improvement approaches, the decoder adopts a local search operator to improve the solution iteratively until a stopping criterion is met. Chen and Tian [43] proposed a NeuRewriter model that consists of a region-picking and a rule-picking policy network. However, its performance is significantly affected by the quality of the initial solution since it only partially improves it. Lu et al. [44] presented a model to pick local operators at each step, which yields superior solutions to NeuRewriter but requires prohibitively longer training time. Wu et al. [45] proposed a self-attention based policy network to refine a solution iteratively with a pairwise local operator. Based on [45], Ma et al. [46] proposed a novel neural architecture based on Transformer to learn representations of node location and their position in a sequence separately. More learning based methods for routing problems can be found in [15]. In this paper, we present a DRL based method to learn a construction approach for URPMCS, which exhibits better performance than classic heuristic methods and other learning based ones.

III. Problem Formulation

A. Formulation as Mathematical Programming

An URPMCS instance comprises a depot, z charging stations, and n target points. It is assumed that those charging stations have been set up in the region of interest. The quantity and locations of targets and charging stations are also predefined. The UAV can be recharged at any station as long as it is able to reach there. Since more than one charging stations exist, each of them may be visited by the UAV multiple times or never. We also assume that once the UAV visits a charging station, it will be fully recharged. Each target must be monitored (or visited) exactly once by the UAV, which starts from and finally returns back to the depot.

We use the directed graph $G = (V, A)$ to represent an URPMCS instance, where the node set V includes the depot, the target set $N = \{1, 2, \cdots, n\}$ and the charging station set $C = \{1, 2, \cdots, z\}$, and the edge set A contains all edges between nodes in V. The UAV starts and ends at the depot and is fully charged upon departure. The charging stations are always available and allow the UAV to be fully charged each time. Therefore, the UAV must fly to the nearby charging station when the power cannot support its remaining monitoring task. Note that the depot only serves as the origin and final destination of the UAV, and does not provide charging service en route. We use vertices 0 and $n + 1$ to denote the depot, representing the origin and destination, respectively. To distinguish the individual visits to the same charging station, we introduce a unique node for each potential visit. We assume that there are additional n_k-1 dummy nodes for charging station k who share the same location as the original one. Therefore, n_k nodes exist for charging station k. Consequently, in the resulting augmented graph, the total number of nodes in V is calculated as,

TABLE II: Variable and parameter definitions

Variable	Description
$0, n+1$	The depot.
N	The set of target points.
C	The set of charging stations.
k	The index for charge stations, $k \in C$.
C_k	The set of dummy nodes added to the graph G corresponding to potential visits to the charging station, $k \in C$.
C'	The set of potential visits to charging stations, $C' = C_0 \cup C_1 \cup \cdots \cup C_z$.
V'	The set of target points and visits of charging stations; $V' = N \cup C'$.
V'_0	The set of target points and visits of charging stations including the depot 0; $V_0 = N \cup C' \cup \{0\}$.
V'_{n+1}	The set of target points and visits of charging stations including the depot $n+1$; $V_0 = N \cup C' \cup \{n+1\}$.
d_{ij}	Distance from node i to node j, where a node can be a target point, the depot, or a charging station.
Q	The battery capacity.
e_u	Energy consumption of the UAV lifting from the depot or charging stations.
e_d	Energy consumption of the UAV landing on the depot or charging stations.
e_h	Energy consumption of the UAV hovering and collecting information at target points.
e_p	Propulsion energy consumption of the UAV alofting and flying.
V_{mr}	The maximum-range speed.
r	The UAV energy consumption per unit traveling distance with V_{mr}.
L_{max}	The maximum flight range of the UAV.
x_{ij}	Binary decision variable: if the UAV flies from node i to node j, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.
y_i	Decision variable specifying the remaining battery capacity when the UAV arrives at node i.
u_i	The position of node i in the UAV route.

$$tol_n = n + \sum_{k=0}^{z} n_k + 2. \tag{1}$$

A feasible UAV route starts from the depot 0, passes through all target nodes and a number of charging stations (might be more than once for each), and ends at the depot $n+1$. Following the custom [47], we assume that the UAV ascends vertically at the depot or charging stations to the desired altitude, then travels at a constant speed, and finally descends vertically to the depot or charging stations. We assume that lifting, landing and hovering related power are constant, which are denoted as e_u, e_d and e_h, respectively. Inspired by the work [48], we assume that the UAV flies at the maximum-range speed V_{mr} (i.e., the optimal UAV speed that maximizes the total traveling distance given the onboard energy Q). r and L_{max} represent the UAV's energy consumption per unit traveling distance and maximum flight range with V_{mr}, respectively. Therefore, rL_{max} is equal to the propulsion energy consumption of the UAV, denoted as e_p. All used parameters and variables are listed in Table II. Accordingly, the mathematical formulation of URPMCS is described as follows:

$$\min \sum_{i \in V'_0, j \in V'_{n+1}, i \neq j} d_{ij} x_{ij}, \tag{2}$$

$$\text{s.t.} \sum_{j \in V'_{n+1}, i \neq j} x_{ij} = 1, \forall i \in N, \tag{3}$$

$$\sum_{j \in V'_{n+1}, i \neq j} x_{ij} \leq 1, \forall i \in C', \tag{4}$$

$$\sum_{j \in V'_0, i \neq j} x_{ji} - \sum_{j \in V'_{n+1}, i \neq j} x_{ij} = 0, \forall i \in V', \tag{5}$$

$$0 < y_j \leq y_i - (rd_{ij} + e_h) x_{ij} + Q(1 - x_{ij}), \quad \forall i \in N, j \in N, i \neq j, \tag{6}$$

$$0 \leq y_j \leq y_i - (rd_{ij} + e_d) x_{ij} + Q(1 - x_{ij}), \quad \forall i \in N, j \in C' \cup \{n+1\}, i \neq j, \tag{7}$$

$$0 < y_j \leq Q - (rd_{ij} + e_h + e_u) x_{ij}, \forall i \in C' \cup \{0\}, \quad \forall j \in N, i \neq j, \tag{8}$$

$$0 \leq y_j \leq Q - (rd_{ij} + e_u + e_d) x_{ij}, \forall i \in C' \cup \{0\}, \quad \forall j \in C' \cup \{n+1\}, i \neq j, \tag{9}$$

$$u_i - u_j + 1 \leq tol_n (1 - x_{ij}), \forall i \in V', j \in V'_{n+1}, i \neq j, \tag{10}$$

$$1 \leq u_i \leq tol_n, i \in V'_{n+1}, \tag{11}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V'_0, i \neq j. \tag{12}$$

We consider the objective of minimizing the traveled distance as defined in formula (2). Constraint (3) enforces that each target node is visited exactly once. Constraint (4) ensures that each charging station (including the dummy nodes) is visited once at most. Constraint (5) guarantees that the number of incoming edges at each node in set V' (i.e., the set of target points and visits of charging stations) equals the number of outgoing edges. Constraints (6-9) keep the UAV battery always above 0 and ensure the UAV satisfies the energy restriction. For example, constraint (6) restricts the remaining battery capacity of the UAV at target node i to equal the sum of UAV's battery capacity at target node j and the energy consumption of flying from target i to target j if the decision variable $x_{ij} = 1$. Constraints (10) and (11) are used for subtour elimination [49]. Finally, constraint (12) defines the domain of binary variables.

B. Formulation as Markov Decision Process

The route construction in URPMCS can be viewed as a sequential decision process, which can be naturally modeled as a Markov decision process (MDP) and solved using RL accordingly. We define the elements of MDP, including the state space, the action space, the transition rule, and the reward function as follows.

State: The state $s_t = (N_t; Q_t)$ consists of the partial solution N_t and the remaining battery capacity Q_t at time step t. N_t contains all visited nodes until time step t. At time step 0, N_0 represents the depot and Q_0 is equal to the maximum battery capacity Q.

Action: The action space at time step t is represented as $A_t = (V_t; C_t)$, where V_t denotes the set of available targets who have not been visited but could be reached by the UAV at time step t, and C_t denotes the set of available charging stations which the UAV could reach at time step t. The RL

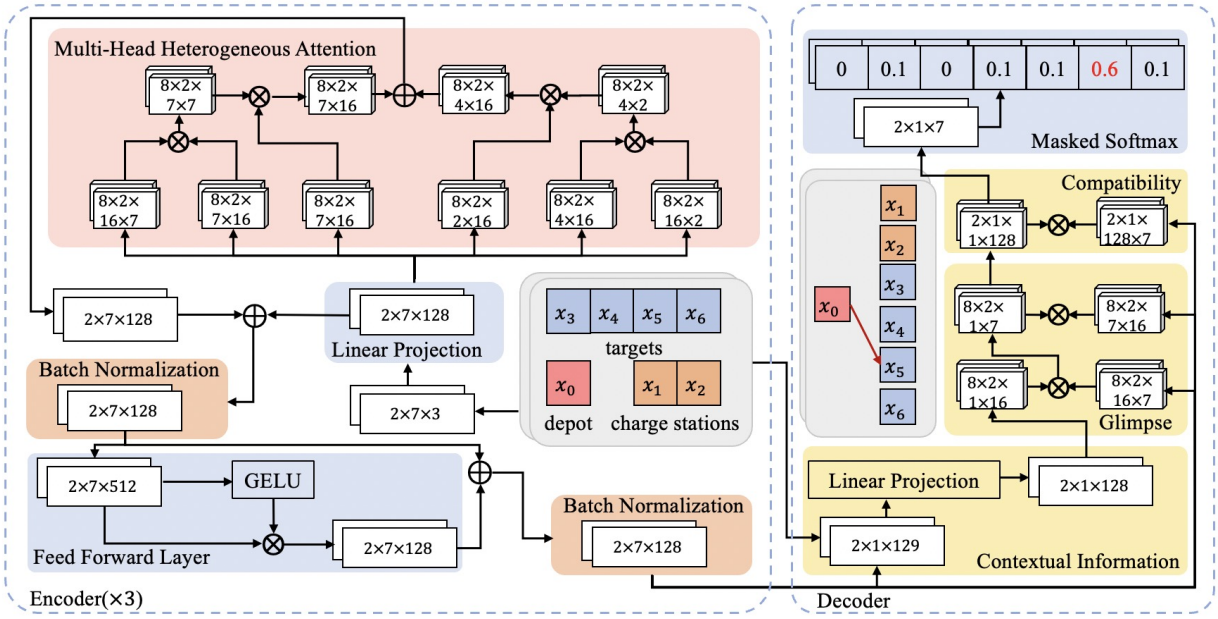


Fig. 2: Illustration of our neural network for two instances with a depot, two charging stations, and four target nodes. The neural network consists of an encoder with three attention layers and a decoder with a multi-head attention layer and a single head attention layer.

agent needs to select an action a_t from A_t at time step t unless meeting the terminal state.

Transition: The next state $s_{t+1} = (N_{t+1}; Q_{t+1})$ is determined by the node chosen at time step t. The partial solution is concatenated with the newly selected node o_j, i.e., $N_{t+1} = (N_t; \{o_j\})$. If o_j is a target node, $Q_{t+1} = Q_t - q_{ij}$, where q_{ij} denotes the power consumed by the UAV for flying from last node o_i to o_j. If o_j is a charging station, then $Q_{t+1} = Q$. If all targets have been visited and the newly selected node is the depot, the next state s_{t+1} is the terminal state.

Reward: To minimize the total travel distance of the UAV, we define the reward as the negative of the objective value. Hence, the reward is the accumulation of the negative travel distances of all intervals, denoted as $R = -\sum_{t=1}^{T} d_{ij}$ where d_{ij} is the distance between node i and node j selected at time step $t-1$ and time step t.

IV. Methodology

In this section, we present a heterogeneous attention based network to parameterize and learn the agent policy p_θ, which automatically selects a node at each time step. The final output solution of the policy network is a permutation of nodes consisting of all targets, the depot and (partial or full) charging stations, described as $\pi = \{\pi_0, \pi_1, \cdots, \pi_T\}, \pi_t \in V$. The probability of a solution is determined by the learnt policy p_θ, described as follows,

$$P(\pi \mid \lambda) = \prod_{t=0}^{T} p_\theta(\pi_t \mid \lambda, \pi_{1:t-1}), \tag{13}$$

where λ is the current problem instance and T is the time step limits for route construction.

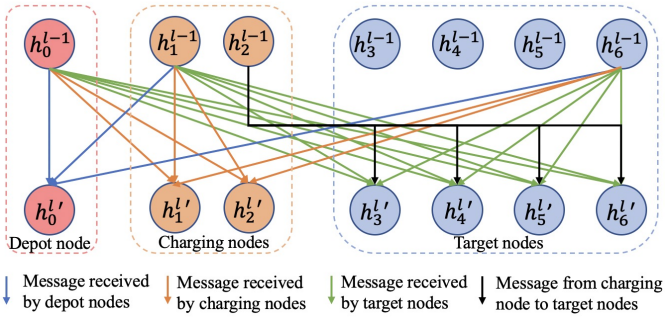


Fig. 3: Illustration of our multi-head heterogeneous attention for an instance with a depot, two charging stations and four target nodes.

A. Heterogeneous Attention based Policy Network

URPMCS is a node selection problem. The input to the problem is an unordered sequence of nodes, and the solution to the problem is a sequence of nodes permuted in order. The encoder-decoder architecture is a sequence-to-sequence paradigm and can predict decisions directly from the input. In this way, the expensive search space exploration is avoided, and the complexity of an NP-hard problem solver becomes polynomial to the problem size [45]. A solution can then be returned instantly for real-time decision-making. Therefore, we design an encoder-decoder-like policy network to learn the policy p_θ, which is depicted in Fig. 2. Our HADRL neural network exploits an architecture akin to Kool et al. [37], and extend it with a heterogeneous attention mechanism and a modified Feed-forward (FF) layer. Since there are two primary types of nodes, i.e., targets and charging stations, the heterogeneous attention is responsible for learning the respective relations between them.

1) Encoder: We exploit a multi-head heterogeneous attention mechanism to learn representations. We encode the input

node sequence $X = \{o_1, o_2, \cdots, o_i, \cdots, o_{n+z+1}\}$ to yield the initial node embeddings h_i^0 with dimension $d_h = 128$ through a linear projection, where, 1) o_i is a 3D vector, including 2D coordinates of the node and a binary variable din, which equals 1 if the node is a charging station, otherwise 0; 2) i is the node index. Then the initial node embeddings are transformed into advanced node embeddings h_i^L through L attention layers, each of which consists of a multi-head heterogeneous attention sublayer and a modified FF sublayer.

Multi-head Heterogeneous Attention Sublayer. The original multi-head attention (MHA) uses self-attention to learn the relationship between two arbitrary nodes in the input sequence. However, different from classic routing problems such as TSP or CVRP, which mainly involve one type of nodes excluding the depot, the nodes in URPMCS have heterogeneous roles. Thus, the relations between nodes in URPMCS are more complex, requiring a more subtle network to learn the representations. To tackle this issue, we rely on two types of attentions, i.e., self-attention and cross-attention for each target node to identify the relations to all charging nodes, as depicted in Fig. 3. The details of the heterogeneous attention mechanism are introduced below.

To extract features, self-attention calculates three vectors, i.e., *query*, *key* and *value*, based on the input sequence. Let h_i^{l-1} denote the node embedding of the attention layer l-1 ($l \in L$), and d_q, d_v denote the dimensions of query/key and value, respectively, where $d_q = d_v = \frac{d_h}{M}$ and $M = 8$ refers to the number of heads. Query, key and value are expressed as follows,

$$q_i = W_q h_i^{l-1}, k_i = W_k h_i^{l-1}, v_i = W_v h_i^{l-1}, \tag{14}$$

where q_i, k_i and v_i refer to query, key and value of node i, respectively, and $W_q, W_k \in R^{d_h \times d_q}$ and $W_v \in R^{d_h \times d_v}$ are trainable parameters. The compatibility between node i and node j is computed as the scaled dot product of the query of node i and key of node j, and then further processed by a softmax function to obtain the node weight as follows,

$$\mu_{ij} = \frac{q_i \top k_j}{\sqrt{d_q}} \tag{15}$$

$$a_{ij} = \frac{e^{\mu_{ij}}}{\sum_{j'} e^{\mu_{ij'}}}. \tag{16}$$

At the same time, another attention mechanism is leveraged to learn the relationship between charging nodes and target nodes. Suppose that h_i^t is the embedding of the ith target node, and h_j^c is the embedding of the jth charging node, where we omit l-1 for simplicity. Accordingly, the query, key and value in this attention are computed as follows,

$$q_i^t = W_{qt} h_i^t, k_j^c = W_{kc} h_j^c, v_j^c = W_{vc} h_j^c, \tag{17}$$

where all parameter matrices are trainable. Hence, the node weights between charging nodes and target nodes are calculated as follows,

$$\mu_{ij}^{tc} = \frac{q_i^t \top k_j^c}{\sqrt{d_q}} \tag{18}$$

$$a_{ij}^{tc} = \frac{e^{\mu_{ij}^{tc}}}{\sum_{j'} e^{\mu_{ij'}^{tc}}}. \tag{19}$$

The single head vector h_i^m is the sum of the heads from all types of attention as follows,

$$h_i^m = \sum_j a_{ij} v_j + \sum_j a_{ij}^{tc} v_j^c. \tag{20}$$

Note that the attention heads from target nodes to charging nodes only contribute to the target node embeddings, while the heads added to the charging nodes embeddings are all zeros. Afterwards, the multi-head vector concatenates different messages from different heads, and then passes through a skip-connection layer and a batch normalization (BN) layer [50] as follows,

$$h_i^{l'} = BN\left(h_i^{l-1} + W_{out} Cat\left(h_i^1, \cdots, h_i^M\right)\right), \tag{21}$$

where W_{out} is a trainable parameter and Cat is the concatenation operator.

Feed-forward Sublayer. In the original attention model [37], the feed-forward (FF) sublayer takes a vector as the input and passes it through two learned linear transformations, between which a rectified-linear (ReLU) activation function is applied. In this paper, we exploit a new FF network, which employs a Gaussian Error Linear Unit (GELU) to replace the ReLU activation function as follows,

$$FF\left(h_i^{l'}, W_1, W_2\right) = GELU\left(h_i^{l'} W_1\right) W_2, \tag{22}$$

where W_1 and W_2 are trainable parameters. As proved to be of high-performance for neural network [51], the GELU activation function is defined as $x\Phi(x)$, where $\Phi(x)$ is the standard Gaussian cumulative distribution function [52].

The output vector of the modified FF sublayer then passes through a skip-connection and a BN layer as follows,

$$h_i^l = BN\left(h_i^{l'} + FF\left(h_i^{l'}, W_1, W_2\right)\right). \tag{23}$$

The advanced node embeddings are finally obtained by passing through L attention layers. Subsequently, the graph embedding is computed to represent the global graph information as follows,

$$\bar{h} = \frac{1}{n+z+1} \sum_{i=0}^{n+z+1} h_i^L. \tag{24}$$

Both the advanced node embeddings and the graph embedding are used as inputs to the decoder.

2) Decoder: The decoder outputs a probability distribution over the input nodes at each decoding step based on the graph embedding and node embeddings produced by the encoder. A vector h_t^{con} representing the context is also computed at each decoding step, which combines the graph embedding $\bar{h}$, the node embedding of the last visited node h_{t-1}^L and the remaining battery capacity Q_t as follows,

$$h_t^{con} = \bar{h} W_{graph} + Cat\left(h_{t-1}^L, Q_t\right) W_{con}, \tag{25}$$

where W_{graph} and W_{con} are trainable parameter matrices. In addition, at the first decoding step, the last node embedding is replaced with trainable parameters.

The decoder is made of a MHA layer and a single head attention layer. A glimpse h_t^g aggregates the messages from different parts of nodes through a MHA layer as follows,

$$h_t^g = MHA\left(h_t^{con}, W_k^g h^L, W_v^g h^L\right), \tag{26}$$

where W_k^g and W_v^g are trainable parameter matrices, and h^L is the node embeddings output by the encoder.

The query and key vectors in the single head attention are computed following Eq. (27) where W_Q and W_K are trainable parameter matrices. Then, the compatibility between the query and the key is calculated following Eq. (28), where C_p is the clip range for better exploration (10 in practice).

$$q_t = W_Q h_t^g; k_i^t = W_K h_i^L, \tag{27}$$

$$h_i^t = C_p \cdot tanh\left(\frac{q_t \top k_i^t}{\sqrt{d_q}}\right). \tag{28}$$

Meanwhile, invalid nodes are masked at each decoding step, by setting the compatibility of those nodes to negative infinity. There are four kinds of invalid nodes in URPMCS, i.e., 1) target nodes that have been already visited; 2) nodes that the UAV cannot reach at the current step due to the insufficient battery capacity; 3) target nodes that would cause the UAV to fail to reach any charging station at the next step if the UAV visits them at the current step; and 4) the depot when there are still remaining target nodes that need to be visited.

Afterwards, we compute the probability of the node to be visited at the next step by a softmax function as follows,

$$P\left(\pi_t \mid \lambda, \pi_{1:t-1}\right) = softmax\left(h^t\right). \tag{29}$$

The decoding process is iteratively performed until all target nodes are covered and the UAV returns to the depot.

A potential method that lets the network learn to avoid allocating probabilities to invalid nodes by itself is to add a penalty term to the reward. Specifically, we can give bad actions that choose an invalid node a negative reward as a penalty term. Therefore, infeasible solutions violating constraints will have low rewards; inversely, feasible paths will have a relatively large reward. After dozens of training episodes, the network may learn a policy that allocates low probabilities to invalid nodes.

In this paper, we leverage two types of decoding strategies, i.e., a *greedy* strategy that always chooses the node with the maximum probability, and a *sampling* strategy that samples nodes according to the probability. We use the sampling strategy in the training process for better exploration. In the inference process, the performances of the two strategies are both investigated.

B. Training

The training algorithm we adopt in our method (i.e. HADRL) is summarized in Algorithm 1, in which the REINFORCE [53] is integrated with a rollout baseline [37]. Our HADRL encompasses two neural networks, i.e., a policy network p_θ which outputs a probability distribution over the action space given the current state, and a baseline network b_φ (the rollout baseline) which calculates the baseline reward by selecting the node with the greedy strategy to reduce variance. The baseline network shares an identical structure to the policy network. After dealing with a batch of instances and receiving the total rewards, the parameters θ of the policy network are updated using the policy gradient algorithm as follows,

ALGORITHM 1: Training for HADRL

Input: Training dataset I, number of epochs E, steps per epoch T, batch size B; policy network p_θ with parameters θ; baseline network b_φ with parameters φ
Output: A well-trained model p_θ.

Initialize policy network parameters θ and baseline network parameters φ;
for *epoch = 1, 2, ⋯, E* **do**
  Sample B instances from dataset I
  for *instance = 1, 2, ⋯, B* **do**
    Set initial state of instance $s_0, t \leftarrow 0$;
    while $t < T$ **do**
      $P\left(\pi_t \mid \lambda, \pi_{1:t-1}\right) \leftarrow$ the policy network p_θ;
      select an action π_t with the sampling strategy;
      Obtain the reward r_t and transit the next state s_{t+1};
      $t = t + 1$;
    end
    $R_i = \sum_{t=1}^{T} r_t$;
    $b_\varphi\left(\lambda\right) \leftarrow$ the reward obtained by the baseline network with the greedy strategy;
  end
  Update θ according to Eq. (31);
  if $OneSidePairedTTest\left(p_\theta, b_\varphi\right) < \alpha$ **then**
    $b_\varphi = p_\theta$
  end
end

$$d_\theta = \frac{1}{B}\sum_{\lambda=1}^{B}\left(R\left(\lambda\right) - b_\varphi\left(\lambda\right)\right) \triangledown_\theta \log p_\theta\left(\pi_\lambda\right), \tag{30}$$

$$\theta_{t+1} = \theta_t + \alpha d_\theta. \tag{31}$$

where d_θ is the gradient of parameters of the policy network; B is the batch size; $R(\lambda)$ is the reward of instance λ calculated by the policy network; $b_\varphi(\lambda)$ is the baseline reward of instance λ calculated by the baseline network and $p_\theta(\pi_\lambda)$ is the probability of determining the solution π_λ by the policy network.

Furthermore, a paired t-test [54] is conducted at each epoch to determine whether the parameters of the baseline network should be updated. If the policy network is significantly superior to the baseline network, the latter parameters will be replaced with the former. After dozens of training steps in each epoch, the trained policy network is able to yield high-quality solutions.

We analyze the complexity of the proposed HADRL method referring to the work in [55]. The complexity of the proposed HADRL is $O(n_e s_p n_\theta C_g)$, where n_e is the number of epochs; s_p is the size of epoch divided by the batch size; n_θ is the number of elements of the parameter vector, and C_g is the time complexity of calculating the gradient of each element in the parameter vector θ.

C. Curriculum Learning Strategy

To further improve the generalization performance of our HADRL, we propose a curriculum learning strategy. It is

simple yet effective by increasing the difficulty of the learning tasks in a gradual manner [56]. We found that the problems with fewer charging stations are harder for the proposed HADRL than those with more charging stations, given the same number of target nodes. Hence, we gradually increase the difficulty of the learning task by reducing the number of charging stations. Specifically, we evenly reduce the number of charging stations as epochs increase, where we empirically reduce one charging station per 10 epochs. For instance, at the beginning, there are 100 target nodes with 10 charging stations, and there will be only one charging station left in the end. Doing so allows the DRL agent to perform favorably and robustly for problems with various numbers of charging stations as it has learnt relevant knowledge properly during training.

V. Experiment

In this section, we conduct extensive experiments to verify the effectiveness of the proposed HADRL method at solving URPMCS. The experiments are performed for a rotary-wing UAV flying at a fixed height. Specifically, a UAV departs from the depot to monitor the targets by visiting them sequentially, goes to charging stations when the battery is going to run out, and then returns to the depot when it has covered all targets. The objective is to minimize the total distance traveled by the UAV. Note that the URPMCS is a NP-hard problem whose theoretical computation time (or runtime) grows exponentially as the problem size scales up.

A. Experiment Setting

We describe the data generation method for the training dataset and test dataset and the hyperparameters setting for our HADRL model. Specifically, in URPMCS, the depot and targets are uniformly sampled within the unit square $[0, 1]^2$ following the conventions in [13], [32], [37], [41]. The charging stations are randomly sampled from the discrete set $[0, 0.25, 0.5, 0.75, 1] \times [0, 0.25, 0.5, 0.75, 1]$. In practice, we ignore the hovering-related power e_h at monitoring nodes, as we assume that the time needed for it to hover there and collect information is negligible. Moreover, since the UAV only needs to take off and land at charging stations, and the related power costs(i.e., e_u and e_d) are constant, we only consider the remaining power (i.e., the battery capacity Q minus e_u and e_d), i.e., the energy used to keep UAV alofting and flying, termed as the propulsion energy consumption e_p. In addition, we set r to 1 for simplicity. Therefore, the maximum flight range L_{max} of the UAV, which is set to 3 in our experiments, mostly depends on the propulsion energy consumption e_p. To comprehensively verify the performance, we evaluate our HADRL on instances with different problem sizes, i.e., 20 targets with 2 charging stations, 50 targets with 5 charging stations, and 100 targets with 10 charging stations, and term them as T20C2, T50C5, T100C10, respectively. Moreover, to assess its generalization capability, we further test our method on larger problem sizes and different distributions. We also conduct an ablation study to justify the effectiveness of the designed heterogeneous attention and GELU activated FF sublayer. Finally, we investigate the impact of different numbers of charging stations on the performance of our curriculum learning strategy.

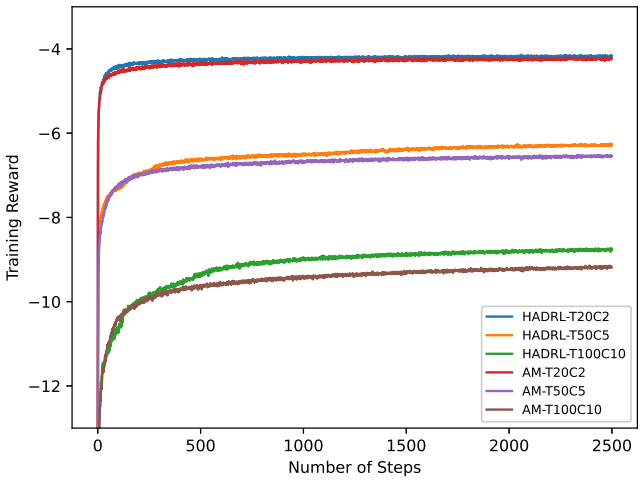


Fig. 4: Training reward curves.

During training, the hyperparameters are shared among all problem sizes. The training instances are randomly generated on-the-fly, where 1024 instances are generated for each batch, with 1250 batches in each epoch. We empirically set the epoch size to 100, since we found that extra epochs may only lead to slight improvement in performance but much longer training time. The dimensions of the node embeddings and the hidden layers in the encoder and decoder are both set to 128. In addition, we set the number of attention layers of the encoder to 3. Similar to [37], we train our HADRL using the Adam optimizer with learning rate 10^{-4}, decaying rate 0.995, and update the learning rate as follows:

$$decayed\_lr = lr * (decaying\,rate)^{n\_epoch} \tag{32}$$

where $decayed\_lr$ and lr are the learning rates for the next and current epoch, respectively, and n_epoch is the current number of epochs. In addition, the indicator α in the paired t-test is set to 0.05. Regarding the training cost, each epoch consumes an average training time of 7.13min (minutes), 11.03min (with single 3090 GPU) and 21.49min (with two GPUs) for problem T20C2, T50C5 and T100C10, respectively. Pertaining to the validation, 10000 instances are generated following a uniform distribution for all problem sizes. All experiments are executed on a server equipped with 8 RTX 3090 GPUs, of which we use two. Our implementation code is publicly available[1].

B. Comparison Analysis

With respect to the proposed HADRL method, we adopt two versions of the decoding strategy, i.e., Greedy and Sampling. Regarding the latter, we sample $\mathbb{N}$ solutions for each instance and return the best one, where we set $\mathbb{N}$ to 1280 and 12800, termed as HADRL(1280) and HADRL(12800), respectively. To evaluate the effectiveness of our HADRL method against others, we adopt various representative methods as baselines, which include:

[1]https://github.com/mingfan321/HADRL

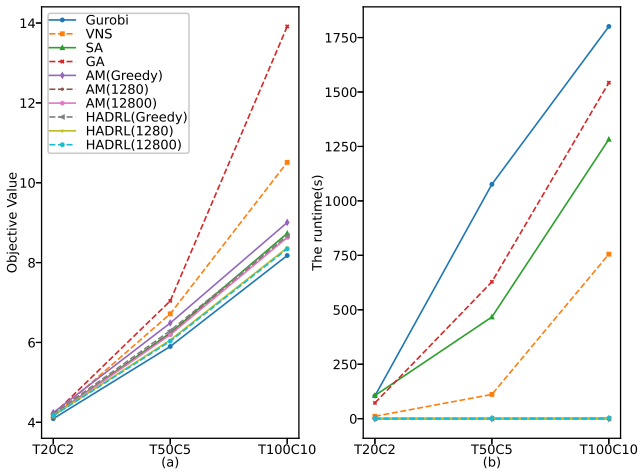


Fig. 5: Diagrams for all algorithms on URPMCS instances of different sizes. (a) Tour length. (b) Runtime.

1) Gurobi, a state-of-the-art exact solver for combinatorial optimization problems;
2) Variable Neighborhood Search (VNS), a heuristic method based on neighborhoods;
3) SA, a meta-heuristic method with guided neighborhood search;
4) GA, a meta-heuristic method based on population;
5) Attention Model (AM), a learning based method adapted to solve URPMCS. The hyperparameters of AM are the same as our HADRL.

Before conducting comprehensive comparisons, we first plot the rewards training curves for our HADRL and the AM method for all cases in Fig. 4. These curves show that the rewards increase sharply for both methods at the beginning, after which our HADRL eventually converges to a higher reward than that of the AM. The reward becomes smaller as the number of targets increases, which indicates longer total distance traveled by the UAV, since the reward is negative. Given that the computation complexity grows exponentially as the URPMCS scales up, we allow reasonably long time limit for Gurobi when solving T100C10, i.e., 1800 seconds. Note that Gurobi needs less than 1800 seconds to compute optimal solutions for T20C2 and T50C5. All the baselines are implemented in Python.

As the conventional methods, including both exact and heuristic ones, consume much more computation time, we only generate 30 URPMCS instances for each problem size for testing. They follow the same distribution as the training one, which are shared by our HADRL method and all baselines. We record the results of our method and baselines on all sizes of URPMCS instances in Table III, which include average objective value (Obj.), gap, and runtime (Time). The smaller the Obj., the better the solution quality. The gap here is calculated by comparing the objective value of a method with the best one found among all methods to show the advantage of the best algorithm over other algorithms. Moreover, the shorter the runtime, the more computationally efficient the algorithm.

In Table III, Gurobi attains the best solutions for all problem sizes (with considerably longer runtime) among all methods. Hence, the gap of others is calculated by comparing them with Gurobi. We observe that, among all heuristic methods, our HADRL achieves the smallest objective value on T50C5 and T100C10, while VNS performs the best on T20C2. However, VNS is inferior to SA on T50C5 and T100C10. Furthermore, GA obtains the worst solutions for all cases, except for AM (Greedy) on T20C2 where the latter performs slightly inferior to GA. Regarding the learning based methods, our HADRL outperforms AM in all cases with either type of decoding strategy. Both Sample 1280 and Sample 12800 yield smaller objective values and gaps than that of Greedy, with Sample 12800 surpassing Sample 1280. Furthermore, AM (Greedy) performs worse than SA for all cases while AM(12800) performs much better, highlighting the ability of the sampling strategy in improving the solution quality.

In terms of computation efficiency, we observe that the runtime of Gurobi and conventional heuristic methods grows almost exponentially as the problem scales up, while that of the learning-based methods grows linearly. In particular, the runtime of Gurobi and conventional heuristic methods is significantly longer than that of the learning-based ones, the latter of which is almost around 1 second. While the overall runtime of our HADRL is slightly longer than that of AM, the Sample 12800 is also longer than that of Sample 1280 for both learning based methods, which are still far shorter than that of the conventional methods. We also plot the trend of the tour length and runtime of all algorithms along different problem sizes in Fig.5, to highlight our advantage on computation efficiency. The runtime of our HADRL only slightly increases with the problem scale, which reveals a desirable trade-off between solution quality and runtime.

C. Generalization Analysis

To assess the generalization performance of our HADRL method, we conduct two types of experiments: 1) apply the model learnt for a problem size to a larger one; 2) apply the model learnt for a uniform distribution to other distributions, i.e., Gaussian distribution and Rayleigh distribution.

Regarding the size generalization, we generate 30 instances of larger sizes, i.e., 150 targets with 15 charging stations and 200 targets with 20 charging stations, and term them as T150C15 and T200C20, respectively. We adopt the model learnt for T100C10 to solve T150C15 and T200C20 and also compare it with Gurobi, VNS, SA, GA and AM. The experimental results are given in Table IV. For each problem size, our HADRL(12800) yields the smallest objective value, while both HADRL (Greedy) and AM (Greedy) already performed better than Gurobi, VNS, SA and GA, where HADRL (Greedy) achieves higher solution quality than that of AM (Greedy). These results show that our HADRL method has a favorable generalization capability on larger problem sizes.

Regarding the distribution generalization, we generate 30 instances in which targets locations are sampled from a Gaussian distribution with different standard deviations and a Rayleigh distribution with different scales for each problem size. The standard deviations adopted in the Gaussian distribution are 1.0, 0.8 and 0.6, while the scales used in the Rayleigh

TABLE III: Comparison results on URPMCS instances of different sizes.

Method	T20C2 Obj.	T20C2 Gap	T20C2 Time	T50C5 Obj.	T50C5 Gap	T50C5 Time	T100C10 Obj.	T100C10 Gap	T100C10 Time
Gurobi	**4.091**	**0.00%**	104s	**5.899**	**0.00%**	1076s	**8.177**	**0.00%**	1800s
VNS	4.155	1.55%	11s	6.717	13.86%	111s	10.511	28.54%	755s
SA	4.174	2.01%	106s	6.239	5.77%	466s	8.732	6.79%	1282s
GA	4.196	2.57%	72s	7.040	19.34%	628s	13.914	70.16%	1542s
AM(Greedy)	4.241	3.65%	**<0.01s**	6.486	9.94%	**0.01s**	9.005	10.12%	**0.02s**
AM(1280)	4.171	1.95%	0.14s	6.217	5.39%	0.20s	8.653	5.82%	0.37s
AM(12800)	4.165	1.81%	0.89s	6.193	4.99%	1.13s	8.628	5.52%	1.79s
HADRL(Greedy)	4.207	2.82%	**<0.01s**	6.295	6.71%	**0.01s**	8.675	6.09%	**0.02s**
HADRL(1280)	4.169	1.91%	0.15s	6.061	2.75%	0.21s	8.372	2.39%	0.39s
HADRL(12800)	4.162	1.73%	0.92s	6.031	2.23%	1.17s	8.344	2.05%	1.86s

The **bold** means the best result in each column, throughout the paper.

TABLE IV: Results for generalization to larger sizes.

Method	T150C15 Obj.	T150C15 Gap	T150C15 Time	T200C20 Obj.	T200C20 Gap	T200C20 Time
Gurobi	11.195	9.81%	1800s	14.187	18.38%	1800s
VNS	14.190	39.18%	2485s	17.718	47.84%	5380s
SA	10.941	7.31%	2412s	13.069	9.05%	3948s
GA	23.023	125.82%	3273s	33.316	177.98%	5471s
AM(Greedy)	10.784	5.77%	**0.03s**	12.589	5.04%	0.05s
AM(1280)	10.380	1.81%	0.63s	12.271	2.39%	0.95s
AM(12800)	10.348	1.49%	1.99s	12.211	1.88%	2.79s
HADRL(Greedy)	10.541	3.38%	**0.03s**	12.313	2.74%	**0.01s**
HADRL(1280)	10.232	0.36%	0.66s	11.997	0.10%	0.97s
HADRL(12800)	**10.196**	**0.00%**	2.16s	**11.985**	**0.00%**	2.03s

TABLE V: Generalization to Gaussian and Rayleigh distributions.

	Method	T20C2(1.0) Obj.	T20C2(1.0) Time	T50C5(1.0) Obj.	T50C5(1.0) Time	T100C10(1.0) Obj.	T100C10(1.0) Time	T20C2(0.8) Obj.	T20C2(0.8) Time	T50C5(0.8) Obj.	T50C5(0.8) Time	T100C10(0.8) Obj.	T100C10(0.8) Time	T20C2(0.6) Obj.	T20C2(0.6) Time	T50C5(0.6) Obj.	T50C5(0.6) Time	T100C10(0.6) Obj.	T100C10(0.6) Time
Gaussian distribution	Gurobi	**4.288**	199s	**5.436**	1538s	**7.230**	1800s	**4.195**	271s	**5.558**	1498s	**7.410**	1800s	**4.149**	395s	**5.563**	1484s	**7.287**	1749s
	VNS	4.349	12s	6.106	105s	8.861	596s	4.268	11s	6.224	103s	8.865	594s	4.215	11s	6.269	102s	8.946	640s
	SA	4.410	99s	5.703	334s	7.527	1052s	4.331	91s	5.836	341s	7.568	1031s	4.252	86s	5.868	345s	7.660	1021s
	GA	4.370	142s	6.192	861s	10.551	1772s	4.299	133s	6.377	742s	10.691	1669s	4.198	132s	6.361	912s	10.867	1770s
	AM(Greedy)	4.561	**<0.01s**	6.130	**0.01s**	8.032	**0.01s**	4.460	**<0.01s**	6.271	**0.01s**	8.075	**0.01s**	4.367	**<0.01s**	6.250	**0.01s**	8.231	**0.01s**
	AM(1280)	4.436	0.16s	5.793	0.22s	7.707	0.41s	4.303	0.14s	5.904	0.19s	7.708	0.37s	4.220	0.15s	5.919	0.23s	7.746	0.44s
	AM(12800)	4.423	0.96s	5.762	1.22s	7.650	1.90s	4.288	0.90s	5.870	1.11s	7.640	1.76s	4.221	0.89s	5.895	1.13s	7.688	1.76s
	HADRL(Greedy)	4.468	**<0.01s**	5.811	**0.01s**	7.988	0.02s	4.447	**<0.01s**	6.031	**0.01s**	7.862	0.02s	4.311	**<0.01s**	6.041	**0.01s**	7.958	0.02s
	HADRL(1280)	4.397	0.15s	5.559	0.20s	7.523	0.38s	4.273	0.15s	5.715	0.20s	7.497	0.38s	4.217	0.15s	5.741	0.20s	7.612	0.38s
	HADRL(12800)	4.390	0.88s	5.538	1.12s	7.467	1.81s	4.267	0.90s	5.697	1.14s	7.471	1.84s	4.214	0.88s	5.721	1.13s	7.572	1.81s

	Method	T20C2(1) Obj.	T20C2(1) Time	T50C5(1) Obj.	T50C5(1) Time	T100C10(1) Obj.	T100C10(1) Time	T20C2(2) Obj.	T20C2(2) Time	T50C5(2) Obj.	T50C5(2) Time	T100C10(2) Obj.	T100C10(2) Time	T20C2(3) Obj.	T20C2(3) Time	T50C5(3) Obj.	T50C5(3) Time	T100C10(3) Obj.	T100C10(3) Time
Rayleigh distribution	Gurobi	**4.347**	265s	**5.467**	1548s	**7.318**	1794s	**4.179**	171s	**5.427**	1532s	**7.315**	1800s	**4.276**	378s	**5.568**	1483s	**7.496**	1800s
	VNS	4.417	11s	6.223	103s	9.042	622s	4.238	13s	6.119	111s	9.048	644s	4.377	12s	6.340	103s	9.259	620s
	SA	4.440	88s	5.743	327s	7.725	1055s	4.258	88s	5.795	340s	7.667	1057s	4.384	96s	5.802	359s	7.780	1141s
	GA	4.425	114s	6.277	725s	11.195	2019s	4.240	144s	6.313	722s	11.210	1940s	4.376	154s	6.443	736s	11.389	1961s
	AM(Greedy)	4.620	**<0.01s**	6.141	**0.01s**	8.321	**0.01s**	4.384	**<0.01s**	6.091	**0.01s**	8.151	**0.01s**	4.578	**<0.01s**	6.225	**0.01s**	8.223	**0.01s**
	AM(1280)	4.435	0.15s	5.804	0.20s	7.879	0.38s	4.263	0.15s	5.736	0.20s	7.746	0.37s	4.423	0.14s	5.888	0.20s	7.856	0.37s
	AM(12800)	4.418	0.87s	5.771	1.11s	7.819	1.77s	4.254	0.88s	5.703	1.12s	7.709	1.77s	4.397	0.85s	5.861	1.11s	7.804	1.76s
	HADRL(Greedy)	4.537	**<0.01s**	5.887	**0.01s**	8.118	0.02s	4.317	**<0.01s**	5.867	**0.01s**	8.038	0.02s	4.543	**<0.01s**	6.041	**0.01s**	8.239	0.02s
	HADRL(1280)	4.437	0.14s	5.618	0.20s	7.697	0.38s	4.231	0.15s	5.593	0.21s	7.686	0.39s	4.408	0.15s	5.721	0.20s	7.796	0.39s
	HADRL(12800)	4.424	0.89s	5.590	1.12s	7.668	1.81s	4.228	0.92s	5.556	1.16s	7.641	1.85s	4.403	0.88s	5.693	1.12s	7.758	1.81s

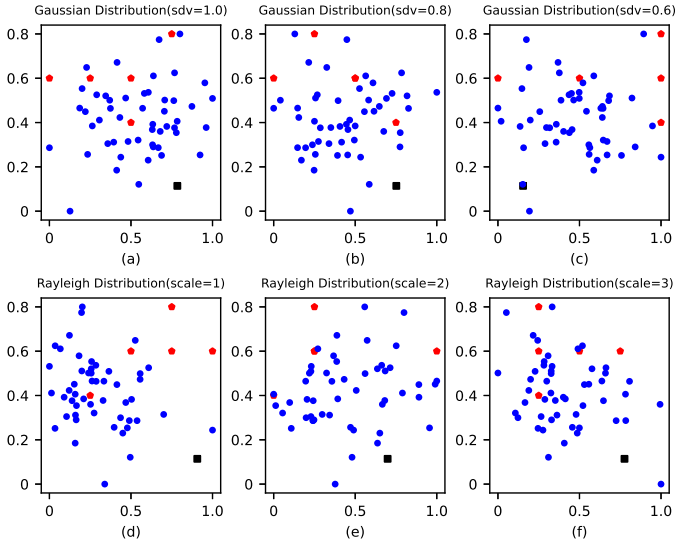


Fig. 6: Instances on different distributions. The black square represents the depot, and red pentagons denote the charging stations while blue circles denote targets.

distribution are 1, 2 and 3. An example instance in each of the above 6 different distributions for T50C5 are displayed in Fig. 6. We apply our model learnt for uniform distribution to solve the instances from different distributions, comparing with Gurobi, VNS, SA, GA and AM. The experimental results for Gaussian distribution are gathered in the upper half of Table V, where T20C2(1.0) refers to the instances sampled from the Gaussian distribution with the standard deviation of 1.0, and others follow the same naming convention. Regarding the Gaussian distribution, HADRL(12800) ranks second which is only slightly inferior to Gurobi on T20C2(0.8), T50C5(1.0,0.8,0.6) and T100C10(1.0,0.8,0.6) in terms of objective values. GA has smaller objective values than HADRL(12800) on T20C2(1.0) and T20C2(0.6), while that of VNS is smaller than HADRL(12800) on T20C2(1.0). Besides, HADRL is superior to AM in all cases with either decoding strategy. Meanwhile, the experimental results for the Rayleigh distributions are gathered in the lower half of Table V, where T20C2(1) refers to instances sampled with scale 1, and others follow the same naming convention. We can observe that Gurobi outperforms all other methods in all cases in terms of objective value. Similar to the Gaussian distribution, the

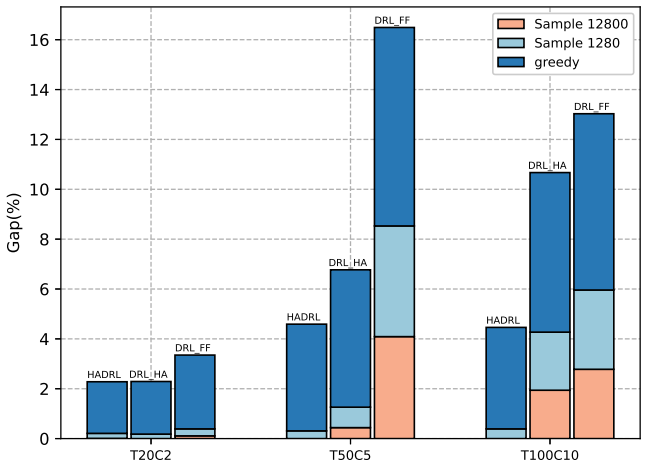


Fig. 7: Gaps of our HADRL variants for URPMCS.

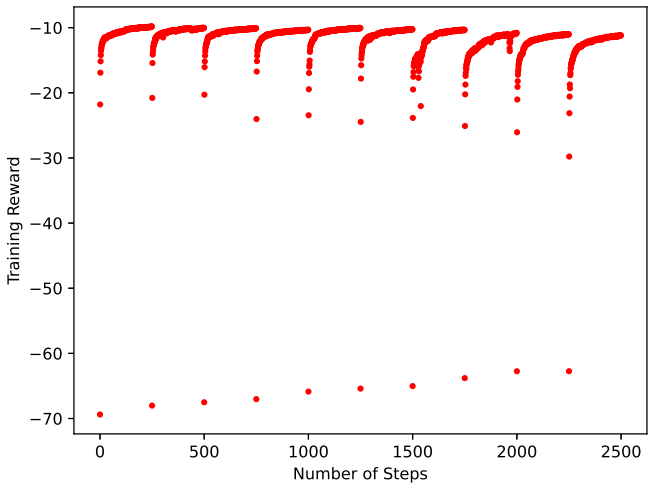


Fig. 8: Training reward curve with curriculum learning.

HADRL(12800) still ranks second on T20C2(2), T50C5(1,2,3) and T100C10(1,2,3). Furthermore, our HADRL(12800) only loses to Gurobi and VNS on T20C2(1). Regarding AM, our HADRL is superior in most cases except for T20C2(1) and T20C2(3). Still, HADRL and AM consume much less runtime than Gurobi and SA. We conclude that our HADRL method has favorable generalization to different distributions in terms of objective value and runtime, especially on larger problem sizes.

D. Ablation Study

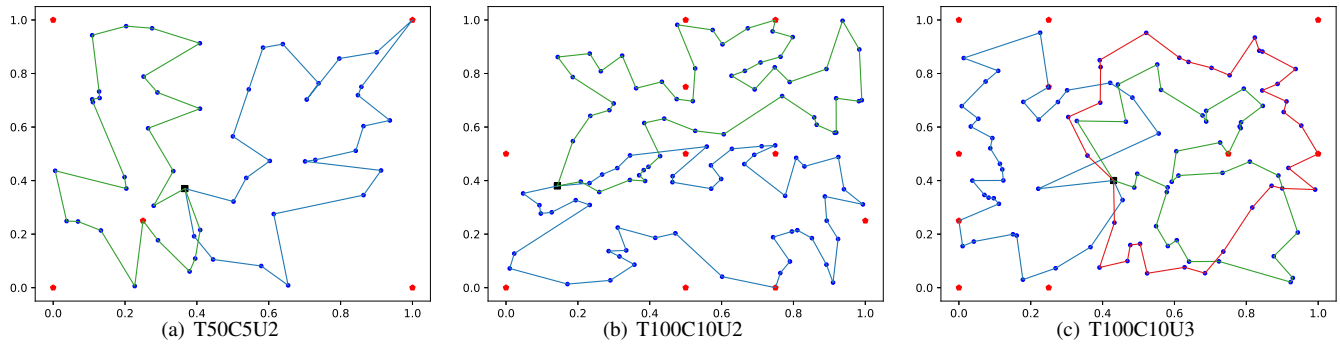
We evaluate the effectiveness of our multi-head heterogeneous attention against the original one termed as DRL_FF on T20C2, T50C5 and T100C10, where DRL_FF primarily follows the AM architecture in [37], but with the modified FF layer. Furthermore, we also evaluate the effectiveness of the modified FF layer against the one equipped with multi-head heterogeneous attention and the FF layer activated by Relu function termed as DRL_HA, on T20C2, T50C5 and T100C10. The gaps of the three models averaged over 1000 instances are displayed in Fig. 7. We can observe that our HADRL is superior to DRL_FF on all problem sizes, which supports the effectiveness of the proposed multi-head heterogeneous attention. On the other hand, while the results are close on T20C2, the gaps of our HADRL are significantly lower than that of DRL_HA on larger problem sizes, which suggests that the modified FF layer is able to improve the performance of our method on larger instances. In addition, the runtime of our HADRL, DRL_FF and DRL_HA here are almost identical to that of HADRL in Table III.

E. Curriculum learning

We assess the performance of our method on problems with fewer charging stations, i.e., 100 targets with one, two, three charging stations, and term them as T100C1, T100C2, T100C3, respectively. We generate 30 instances for each size and adopt Gurobi, VNS, SA, GA and AM as baselines. Furthermore, we still choose the objective value, gap and computation time as the metrics. Experimental results are collected in Table VI. We observe that Gurobi yields the best solutions on T100C2 and T100C3 while SA performs best on T100C1. Our HADRL (12800) performs better than SA on T100C3 and worse on T100C1 and T100C2. Furthermore, the gap between our HADRL(12800) and SA on T100C1 is significantly larger than that on T100C2. We empirically found that the generalizability of our HADRL method deteriorates as the number of charging stations decreases. To improve the robustness, we equip our HADRL with the aforementioned curriculum learning strategy and term it as HACL, the learning curve of which is plotted in Fig. 8. As the charging stations change every 10 epochs, the learning curve drops sharply at the beginning of every 250 training steps (one epoch includes 25 training steps) and rises quickly after a few training steps. We observe that the proposed scheme converges for each size. We also consider three decoding strategies for the developed HACL, i.e., Greedy, Sample 1280 and Sample 12800. The experiment results are recorded in Table VI as well. We observe that the HACL(12800) achieves the smallest objective value on T100C1 and also exhibits competitive performance on T100C2 and T100C3, which suggests that the proposed CL strategy significantly improves the generalizability of HADRL to fewer charging stations. Since HACL and HADRL share the same model structure, they have similar inference times.

TABLE VI: Results with different numbers of charging stations.

Method	T100C1			T100C2			T100C3		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Gurobi	10.508	15.41%	1800s	**8.808**	**0.00%**	1800s	**8.335**	**0.00%**	1800s
VNS	11.029	21.13%	714s	11.109	26.12%	736s	10.673	28.05%	732s
SA	9.180	0.82%	1129s	9.128	3.63%	1393s	8.865	6.37%	1377s
GA	14.668	61.10%	1456s	14.686	66.73%	1450s	14.035	68.39%	1424s
AM(Greedy)	11.564	27.01%	0.02s	10.891	23.65%	0.02s	9.675	16.09%	0.04s
AM(1280)	10.225	12.31%	0.39s	9.741	10.59%	0.37s	9.090	9.07%	0.37s
AM(12800)	10.019	10.04%	1.28s	9.591	8.89%	1.30s	9.019	8.21%	1.30s
HADRL(Greedy)	10.983	20.62%	**<0.01s**	9.971	13.20%	**<0.01s**	9.354	12.22%	**<0.01s**
HADRL(1280)	10.046	10.34%	0.30s	9.396	6.67%	0.30s	8.835	6.01%	0.30s
HADRL(12800)	9.804	7.68%	1.75s	9.302	5.60%	1.50s	8.750	4.98%	1.49s
HACL(Greedy)	9.810	7.74%	**<0.01s**	9.669	9.77%	**<0.01s**	9.327	11.91%	**<0.01s**
HACL(1280)	9.164	0.65%	0.36s	9.100	3.31%	0.36s	8.841	6.07%	0.36s
HACL(12800)	**9.105**	**0.00%**	1.73s	9.019	2.40%	1.76s	8.773	5.26%	1.50s



(a) T50C5U2 (b) T100C10U2 (c) T100C10U3

Fig. 9: Visualization of solutions to multi-UAV instances.

F. Multi-UAV case studies

We extend the proposed method HADRL to a more general problem setting, i.e., Multi-UAV cases in which multiple homogeneous UAVs depart from the same depot to distributedly monitor targets and then fly back to the depot after covering all targets. We consider a shared scenario in which each charging station can support charging multiple UAVs at the same time [47]. We deem the URPMCS in multi-UAV cases as a multi-agent collaborative problem wherein multi-agent work together to maximize the total team reward, and each UAV can be viewed as an agent. The encoders of HADRL models for the single UAV and multi-UAV are the same. Hence, their main difference lies in the decoding procedure. In multi-UAV cases, the decoder follows a one-agent-per-decoding-step routine [57]. Specifically, at each time step t, the decoder first determines which UAV will make a decision, wherein the current UAV index is equal to $t\,\%\,|\mathbb{A}|$, where $\mathbb{A}$ is the set of UAVs and $|\mathbb{A}|$ is the number of UAVs. Moreover, the current context consists of the graph embedding $\bar{h}$, the last node embedding $h^L_{\pi^\alpha_{|\pi^\alpha|}}$ and the remaining battery capacity Q_α of the current agent α. The decoding procedure repeats until all targets are visited and all UAVs return to the depot, the other parts of decoding procedure remaining the same as in the single-UAV case. The decoding procedure for multi-UAV cases is detailed in Algorithm 2.

We evaluate the performance of our HADRL on multi-UAV instances, i.e., 50 targets with 5 charging stations and 2 UAVs, 100 targets with 10 charging stations and 2 UAVs, and 100 targets with 10 charging stations and 3 UAVs, and term them as T50C5U2, T100C10U2, T100C10U3, respectively. The solutions of these three sets of experiments are plotted in Fig.9: (a), (b), and (c), respectively, where the black square represents the depot, and the red pentagons denote the charging stations while blue circles denote targets. We draw lines with different colors to represent different UAV routes. Generally, we observe that the route of each UAV constructed by the HADRL model avoids crossings, indicating that the learned heuristic is able to perform well in multi-UAV cases. Notably, the solutions are obtained in an extremely short runtime, less than 0.1 seconds. Since we consider a shared scenario, several UAVs may reach the same charging station simultaneously in our solutions. We believe that considering communications [58], [59] between UAVs may further foster the performance of the HADRL in multi-UAV cases, e.g., to let each agent know other agents' positions when making decisions. However, it is beyond the scope of this work and we leave it as future work.

ALGORITHM 2: HADRL Decoding Procedure for Multi-UAV Cases

Input: Node embeddings h_i^L, graph embedding $\bar{h}$, set of UAVs $\mathbb{A}$, the remaining battery capacity Q_α of UAV α.
Output: The tours of all UAVs $\pi(=\pi^i,\ldots,\pi^{|\mathbb{A}|})$.

Initialize the tour list of each UAVs, $\pi^\alpha \leftarrow [0]$;
Initialize the remaining battery capacity of each UAV to the maximum battery capacity, $Q_{\alpha,0} \leftarrow Q$;
Set the decision time step t to 0.
while *(not all target are visited)*$\cup$*(any* $\pi^\alpha_{|\pi^\alpha|}! = 0$*)* **do**
  $\alpha \leftarrow t\,\%\,|\mathbb{A}|$;
  $h^{con}_\alpha = \bar{h}W_{graph} + Cat\left(h^L_{\pi^\alpha_{|\pi^\alpha|}}, Q_\alpha\right)W_{con}$;
  $h^g_\alpha = MHA\left(h^{con}_\alpha, W^g_k h^L, W^g_v h^L\right)$;
  $q_\alpha = W_Q h^g_\alpha; k_i = W_K h^L_i$;
  $h^\alpha_i = C_p \cdot tanh\left(\frac{q_\alpha \top k_i}{\sqrt{d_q}}\right)$;
  $h^\alpha_i = h^\alpha_i - \infty *$ (node i is masked);
  $P\left(\pi^\alpha_{|\pi^\alpha|} \mid \lambda, \pi^\alpha_{1:|\pi^\alpha|-1}\right) = softmax\left(h^\alpha\right)$;
  if *is training* **then**
    select an action $\pi^\alpha_{|\pi^\alpha|}$ with the *sampling* strategy;
  else
    select an action $\pi^\alpha_{|\pi^\alpha|}$ with the *greedy* strategy.
  end
  Update the remaining capacity Q_α according to the transition rule.
  Add $\pi^\alpha_{|\pi^\alpha|}$ to the tour π^α.
  $t \leftarrow t+1$.
end

VI. Conclusion

In this paper, we approach the UAV routing problem for traffic monitoring, where multiple charging stations are available to replenish energy for the UAV. To solve this problem, we propose a deep reinforcement learning based method in conjunction with a multi-head heterogeneous attention mechanism to learn a policy for automatic route construction. We also design a curriculum learning strategy to enhance the robustness of our method against different numbers of charging stations. Experimental result shows that the overall performance of our method is superior to existing learning baseline (AM) and

conventional methods. Moreover, the proposed method generalizes well to larger problem sizes, different target distributions and fewer charging stations.

Compared to conventional methods, the advantages of our HADRL method are three-fold: 1) HADRL learns the heuristic for automatically constructing a solution by itself instead of exploiting domain-specific knowledge. 2) HADRL can generate a route for a UAV very fast (in 1 second), while other conventional methods need dozens of seconds or even much longer. 3) HADRL can replan a route for a UAV immediately facing a dynamic situation where some targets may be cancelled, or some charging stations are filled. However, conventional methods must recalculate the route from scratch in such dynamic environments.

Future work will consider more advanced deep architectures for routing, such as POMO [13] to further improve the performance, and test our method on real-world problem instances. Moreover, we will also investigate how to cooperatively route multiple UAVs in the presence of non-shared charging stations.

REFERENCES

[1] M. Li, L. Zhen, S. Wang, W. Lv, and X. Qu, "Unmanned aerial vehicle scheduling problem for traffic monitoring," *Computers & Industrial Engineering*, vol. 122, pp. 15–23, 2018.

[2] A. Puri, "A survey of unmanned aerial vehicles (UAV) for traffic surveillance," *Department of computer science and engineering, University of South Florida*, pp. 1–29, 2005.

[3] H. Liu, X. Li, M. Fan, G. Wu, W. Pedrycz, and P. N. Suganthan, "An autonomous path planning method for unmanned aerial vehicle based on a tangent intersection and target guidance strategy," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[4] H. Liu, X. Li, G. Wu, M. Fan, R. Wang, L. Gao, and W. Pedrycz, "An iterative two-phase optimization method based on divide and conquer framework for integrated scheduling of multiple UAVs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5926–5938, 2020.

[5] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "UAV trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9540–9554, 2021.

[6] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[7] Y. Li, A. H. Aghvami, and D. Dong, "Path planning for cellular-connected uav: A drl solution with quantum-inspired experience replay," *IEEE Transactions on Wireless Communications*, 2022.

[8] Y. Li, A. H. Aghvami, and D. Dong, "Intelligent trajectory planning in uav-mounted wireless networks: A quantum-inspired reinforcement learning perspective," *IEEE Wireless Communications Letters*, vol. 10, no. 9, pp. 1994–1998, 2021.

[9] S. Harwin and A. Lucieer, "Assessing the accuracy of georeferenced point clouds produced via multi-view stereopsis from unmanned aerial vehicle (UAV) imagery," *Remote Sensing*, vol. 4, no. 6, pp. 1573–1599, 2012.

[10] H. Yan, Y. Chen, and S.-H. Yang, "UAV-enabled wireless power transfer with base station charging and uav power consumption," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12883–12896, 2020.

[11] A. H. Fathima and K. Palanisamy, "Optimization in microgrids with hybrid energy systems–a review," *Renewable and Sustainable Energy Reviews*, vol. 45, pp. 431–446, 2015.

[12] S. Erdoğan and E. Miller-Hooks, "A green vehicle routing problem," *Transportation research part E: logistics and transportation review*, vol. 48, no. 1, pp. 100–114, 2012.

[13] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "Pomo: Policy optimization with multiple optima for reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 21188–21198, 2020.

[14] A. Hottung, Y.-D. Kwon, and K. Tierney, "Efficient active search for combinatorial optimization problems," in *International Conference on Learning Representations*, 2021.

[15] B. Li, G. Wu, Y. He, M. Fan, and W. Pedrycz, "An overview and experimental study of learning-based optimization algorithms for vehicle routing problem," *IEEE/CAA Journal of Automatica Sinica*, 2022.

[16] Y. Cao, Z. Sun, and G. Sartoretti, "Dan: Decentralized attention-based neural network to solve the minmax multiple traveling salesman problem," in *International Symposium on Distributed Autonomous Robotics Systems*, 2022.

[17] L. Zhen, M. Li, G. Laporte, and W. Wang, "A vehicle routing problem arising in unmanned aerial monitoring," *Computers & Operations Research*, vol. 105, pp. 1–11, 2019.

[18] V. K. Shetty, M. Sudit, and R. Nagi, "Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles," *Computers & Operations Research*, vol. 35, no. 6, pp. 1813–1828, 2008.

[19] D. W. Casbeer and R. W. Holsapple, "Column generation for a UAV assignment problem with precedence constraints," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1421–1433, 2011.

[20] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *2008 47th IEEE conference on decision and control*, pp. 3953–3958, IEEE, 2008.

[21] A. Thibbotuwawa, G. Bocewicz, P. Nielsen, and B. Zbigniew, "Planning deliveries with UAV routing under weather forecast and energy consumption constraints," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 820–825, 2019.

[22] D. J. Klein, S. Venkateswaran, J. T. Isaacs, J. Burman, T. Pham, J. Hespanha, and U. Madhow, "Localization with sparse acoustic sensor network using UAVs as information-seeking data mules," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 3, pp. 1–29, 2013.

[23] K. P. O'Rourke, W. B. Carlton, T. G. Bailey, and R. R. Hill, "Dynamic routing of unmanned aerial vehicles using reactive tabu search," *Military Operations Research*, pp. 5–30, 2001.

[24] G. W. Kinney, R. R. Hill, and J. T. Moore, "Devising a quick-running heuristic for an unmanned aerial vehicle (UAV) routing system," *Journal of the Operational Research Society*, vol. 56, no. 7, pp. 776–786, 2005.

[25] J. A. Guerrero and Y. Bestaoui, "UAV path planning for structure inspection in windy environments," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1, pp. 297–311, 2013.

[26] H. Savuran and M. Karakaya, "Efficient route planning for an unmanned air vehicle deployed on a moving carrier," *Soft Computing*, vol. 20, no. 7, pp. 2905–2920, 2016.

[27] T. Wen, Z. Zhang, and K. K. Wong, "Multi-objective algorithm for blood supply via unmanned aerial vehicles to the wounded in an emergency situation," *PloS one*, vol. 11, no. 5, p. e0155176, 2016.

[28] G. B. Lamont, J. N. Slear, and K. Melendez, "UAV swarm mission planning and routing using multi-objective evolutionary algorithms," in *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pp. 10–20, IEEE, 2007.

[29] F. Guerriero, R. Surace, V. Loscri, and E. Natalizio, "A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints," *Applied Mathematical Modelling*, vol. 38, no. 3, pp. 839–852, 2014.

[30] K. Sundar and S. Rathinam, "Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 287–294, 2013.

[31] B. N. Coelho, V. N. Coelho, I. M. Coelho, L. S. Ochi, R. Haghnazar, D. Zuidema, M. S. Lima, and A. R. da Costa, "A multi-objective green uav routing problem," *Computers & Operations Research*, vol. 88, pp. 306–315, 2017.

[32] J. Li, L. Xin, Z. Cao, A. Lim, W. Song, and J. Zhang, "Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[33] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[34] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations*, 2015.

[35] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *International Conference on Learning Representations*, 2017.

[36] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[37] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!," in *International Conference on Learning Representations*, 2018.

[38] J. M. Vera and A. G. Abad, "Deep reinforcement learning for routing a heterogeneous fleet of vehicles," in *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pp. 1–6, IEEE, 2019.

[39] J. K. Falkner and L. Schmidt-Thieme, "Learning to solve vehicle routing problems with time windows through joint attention," *arXiv preprint arXiv:2006.09100*, 2020.

[40] L. Xin, W. Song, Z. Cao, and J. Zhang, "Step-wise deep learning models for solving routing problems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4861–4871, 2020.

[41] L. Xin, W. Song, Z. Cao, and J. Zhang, "Multi-decoder attention model with embedding glimpse for solving vehicle routing problems," in *Proceedings of 35th AAAI Conference on Artificial Intelligence*, pp. 12042–12049, 2021.

[42] J. Li, Y. Ma, R. Gao, Z. Cao, A. Lim, W. Song, and J. Zhang, "Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem," *IEEE Transactions on Cybernetics*, 2021.

[43] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[44] H. Lu, X. Zhang, and S. Yang, "A learning-based iterative method for solving vehicle routing problems," in *International Conference on Learning Representations*, 2019.

[45] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems..," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[46] Y. Ma, J. Li, Z. Cao, W. Song, L. Zhang, Z. Chen, and J. Tang, "Learning to iteratively solve routing problems with dual-aspect collaborative transformer," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[47] R. Alyassi, M. Khonji, A. Karapetyan, S. C.-K. Chau, K. Elbassioni, and C.-M. Tseng, "Autonomous recharging and flight mission planning for battery-operated autonomous drones," *IEEE Transactions on Automation Science and Engineering*, 2022.

[48] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing uav," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[49] M. Desrochers and G. Laporte, "Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints," *Operations Research Letters*, vol. 10, no. 1, pp. 27–36, 1991.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, pp. 4171–4186, 2019.

[52] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[53] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[54] H. Hsu and P. A. Lachenbruch, "Paired t test," *Wiley StatsRef: statistics reference online*, 2014.

[55] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3177–3192, 2021.

[56] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, pp. 1–50, 2020.

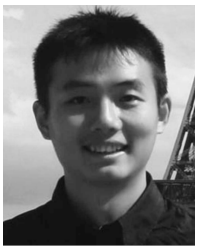
[57] P. Sankaran, K. McConky, M. Sudit, and H. Ortiz-Peña, "Gamma: Graph attention model for multiple agents to solve team orienteering problem with multiple depots," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[58] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10433–10445, 2017.

[59] X. Pang, N. Zhao, J. Tang, C. Wu, D. Niyato, and K.-K. Wong, "Irs-assisted secure uav transmission via joint trajectory and beamforming design," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1140–1152, 2021.

Mingfeng Fan received the B.Eng degree in Transport Equipment and Control Engineering from Central South University, Changsha, China, in 2019, where she is currently pursuing the Ph.D. degree in Traffic and Transportation Engineering. Her research interests include machine learning and UAV path planning.

Yaoxin Wu received the B.Eng degree in traffic engineering from Wuyi University, Jiangmen, China, in 2015, and M.Eng degree in control engineering from Guangdong University of Technology, Guangzhou, China, in 2018. He was a Research Associate with the Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU). He is currently a Ph.D. student with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include combinatorial optimization, integer programming and deep learning.



Tianjun Liao received the bachelor's degree from the National University of Defense Technology, Changsha, China, in 2007, and the Ph.D. degree from the Institute de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle, Universite Libre de Bruxelles, Brussels, Belgium, in 2013. He is currently an Assistant Researcher with Academy of Military Sciences, Beijing, China. His current research interests include heuristic optimization algorithms and automated algorithm configuration.

Zhiguang Cao received the Ph.D. degree from Interdisciplinary Graduate School, Nanyang Technological University, 2017. He received the B.Eng. degree in Automation from Guangdong University of Technology, Guangzhou, China, in 2009 and the M.Sc. degree in Signal Processing from Nanyang Technological University, Singapore, in 2012, respectively. He was a Research Fellow with Future Mobility Research Lab, and Energy Research Institute @ NTU (ERI@N), and a Research Assistant Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore. He is currently a Scientist with the Institute for Infocomm Research (I2R), Singapore. His research interests focus on machine learning for optimization.

Hongliang Guo received his Bachelor of Engineering in Dynamic Engineering and Master of Engineering in Dynamic Control at Beijing Institute of Technology, China, in 2005 and 2007 respectively. He holds a PhD degree in Electrical and Computer Engineering from Stevens Institute of Technology, USA. His research interests include planning and learning under uncertainties, and multi-robot search. He has been an associate professor in University of Electronic Science and Technology of China from 2016 to 2020. In 2021, He joins the Institute of Infocomm Research (I2R) in Agency for Science, Technology And Research (A*STAR), Singapore, as a Scientist.

Guillaume Sartoretti joined the Mechanical Engineering department at the National University of Singapore as an Assistant Professor in 2019. Before that, he was a Postdoctoral Fellow in the Robotics Institute at Carnegie Mellon University. He received his Ph.D. degree in robotics from EPFL in 2016. He also holds a B.S. and an M.S. degree in Mathematics and Computer Science from the University of Geneva. He is interested in the emergence of collaboration/cooperation in large groups of intelligent agents making individual choices based on their local understanding of the world.

Guohua Wu received the B.S. degree in Information Systems and Ph.D degree in Operations Research from National University of Defense Technology, China, in 2008 and 2014, respectively. During 2012 and 2014, he was a visiting Ph.D student at University of Alberta, Edmonton, Canada. He is currently a Professor at the School of Traffic and Transportation Engineering, Central South University, Changsha, China. His current research interests include Planning and Scheduling, Computational Intelligence and Machine Learning. He has authored more than 100 referred papers including those published in IEEE TCYB, IEEE TSMCA and IEEE TEVC. He serves as an Associate Editor of Information Sciences, and an Associate Editor of Swarm and Evolutionary Computation Journal, an editorial board member of International Journal of Bio-Inspired Computation, and Guest Editors of several journals.