JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 20, No. 4, April 2023

Edge-Enhanced Attentions for Drone Delivery in Presence of Winds and Recharging Stations

Ruifan Liu,* Hyo-Sang Shin,† and Antonios Tsourdos‡
Cranfield University, Bedfordshire, England MK43 0AL, United Kingdom

https://doi.org/10.2514/1.I011171

Existing variants of vehicle routing problems have limited capabilities in describing real-world drone delivery scenarios in terms of drone physical restrictions, mission constraints, and stochastic operating environments. To that end, this paper proposes a specific drone delivery problem with recharging (DDP-R) characterized by directional edges and stochastic edge costs subject to wind conditions. To address it, the DDP-R is cast into a Markov decision process over a graph, with the next node chosen according to a stochastic policy based on the evolving observation. An edge-enhanced attention model (AM-E) is then suggested to map the optimal policy via the deep reinforcement learning (DRL) approach. The AM-E comprises a succession of edge-enhanced dot-product attention layers and is designed with the aim of capturing the heterogeneous node relationship for DDP-Rs by incorporating adjacent edge information. Simulations show that edge enhancement facilitates the training process, achieving superior performance with less trainable parameters and simpler architecture in comparison with other deep learning models. Furthermore, a stochastic drone energy cost model in consideration of winds is incorporated into validation simulations, which provides a practical insight into drone delivery problems. In terms of both nonwind and windy cases, extensive simulations demonstrate that the proposed DRL method outperforms state-of-the-art heuristics for solving DDP-Rs, especially at large sizes.

I. Introduction

RAPID developments in drone technologies opened the way for a wide range of civilian applications (e.g., traffic surveillance, medical delivery, and general warehouse), which comprise an important component of the intelligent transport system (ITS). Meanwhile, worldwide authorities have begun to authorize flights beyond the visual line of sight to test the safety of opening up the technology to the wider industry. Integrating drones into ITSs offers increased flexibility and efficiency to transportation, which brings high societal and economic benefits.

Despite their advantages, the introduction of drones as new couriers into the logistics market implies new challenges in terms of logistics planning due to their limited onboard battery and load capacity. The vehicle routing problem (VRP) is a generic optimization class for modeling multivehicle logistics with the objective of seeking the best route for a fleet of vehicles to serve a set of customers [1]. To avoid exceeding the limited flight time of lithium polymer battery-equipped drone couriers, a multitrip VRP (MTVRP) for drone delivery was proposed in Ref. [2], which compensates by making multiple trips to the depot while being subject to budget constraints. However, in the MTVRP, the serving range of delivery is still restricted, and drones continuously return to the depot, resulting in low efficiency. Motivated by this, a new area of research is arising that investigates the placement of recharging stations to extend drone flight distance and maximize covered demand with the least amount of infrastructure investment [3,4], which benefits drone delivery with greater coverage, lower costs, and higher-quality delivery services; and it is ecofriendly by reducing unnecessary return flights. The routing problem considering recharging is termed as the electric vehicle routing problem (EVRP) [5], which is also known as the green vehicle routing problem, and originally proposed for electric cars similarly suffering from limited battery capacity. Furthermore, in addition to the Amazon "beehive" delivery pattern, where all drones pick up parcels from a shared giant center, a delivery request is likely to have its own pickup and delivery locations, such as those in a medical delivery service [6]. This type of route planning problem can be described as a pickup and delivery problem (PDP), which is another variant of the VRP characterized by pairing and precedence relationships. To address the unmanned aerial logistics while considering both recharging facilities and heterogeneous pickup and delivery requirements, a combination of the EVRP and PDP is investigated in this paper. Moreover, compared to ground-based vehicles, unmanned aerial vehicles (UAVs) are severely affected by weather conditions, especially airflow, introducing stochastic factors to the execution environment. Accommodating this uncertainty in planning tends to increase the quality of routes by reducing risks and stabilizing deliveries.

To summarize, the drone delivery problem with recharging (DDP-R) considered in this paper is essentially an optimization problem that finds the best routes for serving pickup and delivery tasks while taking into account load and battery constraints as well as the randomness introduced by weather. Due to its NP (i.e., nondeterministic polynomial time) -hard and stochastic nature, conventional methods including exact and heuristic algorithms struggle to efficiently address this routing problem [7]. Recently, there has been increasing attention on deep reinforcement learning (DRL) for solving combinatorial optimization problems, which has delivered promising results on the basic VRP and some of its variants [8,9]. To this end, this paper explores learning techniques for solving the drone delivery problem considered.

Because most DRL-based approaches are proposed for typical VRPs and have few actual applications for stochastic environments, two key challenges are accordingly identified on the aforementioned drone delivery problem. First, nodes in the DDP-R involve more heterogeneous roles, consisting of pickup tasks, delivery tasks, charging stations, and the depot. Current network architectures are not intended to capture their sophisticated characteristics and connections. Second, routing problems with drones are vulnerable to weather. Performance degradation is likely to happen due to discrepancies between the deterministic planning model and the actual stochastic system in the presence of wind.

To address these challenges, we develop a routing simulator for DDP-Rs in the presence or absence of winds and propose a novel

Presented as Paper 2022-4028 at the AIAA Aviation 2021 Forum, Chicago, IL, June 27–July 1, 2022; received 5 July 2022; revision received 18 October 2022; accepted for publication 28 November 2022; published online Open Access 30 January 2023.

*Ph.D. Candidate, School of Aerospace, Transport and Manufacturing; ruifan.liu@cranfield.ac.uk.

†Professor, School of Aerospace, Transport and Manufacturing; h.shin@cranfield.ac.uk.

‡Professor, School of Aerospace, Transport and Manufacturing; a.tsourdos@cranfield.ac.uk. Senior Member AIAA.

neural network architecture with a stronger ability of description. To summarize, the main contributions are as follows:

1) To the best of our knowledge, this paper is the first attempt to solve the DDP-Rs via the reinforcement learning (RL) method. A good policy is sought by devising the evolution of the environment, masking scheme, and reward function. Through the comparison with state-of-the-art algorithms, the proposed DRL-based planning method provides high-quality solutions (within at most 4% gap to optimal solutions) and is more resilient in the presence of stochastic winds.

2) A policy neural network called the attention model with edges (AM-E) is proposed for solving DDP-Rs, where an edge feature matrix is additionally considered to offer the network greater description power in terms of elaborating node connectivity, without extensively increasing its computational complexity.

3) In contrast to existing studies, this paper also considers the pickup–delivery task configuration, the effect of wind variations, and the presence of charging stations in formulating the drone delivery problem for the most practical purposes. The newly formulated problem has a stochastic nature and complex node relationships that pose difficulties for traditional methods. Therefore, we further describe the problem in the form of a Markov decision process (MDP) [10], which facilitates the generation of AM-E policies.

The rest of the paper is organized as follows: Sec. II provides an overview of related works based on conventional methods and learning techniques. Section III gives an explicit description of the routing problem of concern, including the drone energy consumption model. This is followed by the proposed DRL based in Sec. IV. Section V presents details of the demonstration experiments, in which the experimental settings are described and the evaluation results under deterministic and stochastic cases are presented. Sections VI and VII offers conclusions of this study and outlines a brief plan for future research.

II. Related Works

In this section, we introduce readers to existing works that have addressed vehicle routing problems using conventional methods or using learning-based methods, with a focus on its two variants: pickup and delivery problems with recharging and stochastic routing problems.

A. Exact and Heuristic Methods

Because the VRP is an NP-hard problem, either the EVRP or PDP is a generalization of the VRP; thus, they are equally considered NP-hard in the strong sense [1]. Due to the complexity of the problem, studies using exact approaches are rarely found in the literature; and most of them adopt branch-and-bound algorithms or their variants. A branch-and-cut algorithm was proposed in Ref. [11] to solve the EVRP. Similarly, a branch-price-and-cut algorithm was used in Ref. [12] to solve the EVRP with time windows. Various branch-and-price algorithms were designed, targeting specific extended versions of the EVRP, e.g., with heterogeneous stations, a partial charging policy, and a nonlinear charging rate [13]. In terms of the PDP, extra bounding procedures or column generation schemes are introduced to the basic branch-and-bound algorithm to meet the pickup–delivery constraints [14,15]. It is worth mentioning that based on these methods, some commercial solvers, such as CPLEX and Gurobi, are broadly employed to find the optimal solution for small-sized VRPs and their variants. However, for large-scale problems, exact methods suffer from computational complexity of the problem.

One trend in solving VRPs is to use metaheuristics, with the purpose of tackling the problem approximately but efficiently. Typical metaheuristics include neighborhood search, simulated annealing [16], tabu search, evolutionary metaheuristics [17], and genetic algorithms [18]. Rastani and Çatay [19] integrated an optimal repair procedure in a large neighborhood search heuristic method for solving the load-dependent EVRP with time windows. Another adaptive large neighborhood search method was proposed in Ref. [20] to solve the PDP based on the destroy and recreate principle. Particularly stated as the first solution to the PDP with electric vehicles, a granular tabu search algorithm was developed in Ref. [21] by restricting the neighborhood of the search by creating a promising arc. Although these heuristic methodologies perform well in offline situations, most of them cannot be directly employed in real time or dynamic cases because they fail to efficiently deal with the unforeseen changes, i.e., incoming requests. In terms of stochastic routing problems, the problem is modeled as a stochastic VRP, which is then handled by simulation-based optimization techniques for maximizing the expected objective function. However, the increase of complexity due to stochastic factors makes the problem even harder to be solved in real time.

B. Learning-Based Approaches

In recent years, great potential has been found in employing DRL to resolve routing problems. To use DRL, the route is constructed by appending next visit nodes, formulated as a sequential Markov decision making process. Graph-based policy networks and sequence-to-sequence networks are found in the literature to map the state space to action probabilities.

The pointer network (hereafter referred to as PtrNet) is the first seminar work to cope with the routing problem via learning, which solves the travel sales problem (TSP) via recurrent neural networks [22]. The PtrNet is first implemented in a supervised manner and later extended into an RL framework [23]. Furthermore, the work in Ref. [24] generalizes the application of the PtrNet to a wider range of CO such as the capacitated VRP , where element embeddings could be dynamic. After that, more effective deep learning architectures are proposed to represent problem statements by combining the transformer-based attention model [8,25] or graph-embedded structure [9,26], showing outperforming results in comparison with heuristic methods. A stage has arrived where deep neural networks can effectively extract useful information from customer configurations and obtain high-quality policies for typical routing problems through reinforcement learning.

Extending to variants of the vehicle routing problem, such as the pickup and delivery problem or the electric vehicle routing problem, the aforementioned networks could be directly employed with a redesigned reward scheme and mask policy. However, this is less effective for distinguishing different types of nodes and identifying their relationship for specific variants. To that end, Li et al. [27] propose a heterogeneous attention model for the PDP, in which seven types of attention layers were sophisticatedly designed to consider different roles played by nodes while taking the precedence constraint into account. Lin et al. [26] incorporated the model of Ref. [24] with a graph-embedding component to yield the global information of the graph for the EVRP with time windows. In addition to these efforts, another direction of enhancement is to integrate edge features in the graph neural networks. Instead of a one-dimensional binary adjacent matrix, Gong and Cheng [28] made efforts to exploit more complex edge features, leading to a generic edge-integrated framework. Despite their purpose to boost performance, edge-featured models are not always superior, especially for some node-sensitive tasks [29], which have actually motivated our proposal of the AM-E network.

On the other hand, the routing problem with stochastic model parameters has received increasing attention. Instead of assuming the cost and customers are static and known a priori, taking into account the randomness of planning parameters leads to a higher quality of the solution. Because RL provides promising tools to optimize policies with stochastic state transitions, Bono [30] developed an online representation of problem states enabling real-time planning based on the latest observation of vehicles. Machine learning techniques have also been used to build a probabilistic energy consumption model [31], achieving more energy savings and reliability for routes.

III. Problem Statement

This section describes the drone delivery problem with recharging, presents a stochastic drone energy consumption model in

consideration of varying wind conditions, and then formulates DDP-Rs in the form of MDPs [10].

A. Drone Delivery Problem with Recharging

Suppose a drone delivery scenario, shown as Fig. 1, in which there are n customer requests and each of them is decomposed into a pickup task $i \in \mathbb{P}$ and a delivery task $i+n \in \mathbb{D}$. A drone is sent from the depot point with a fully charged battery and finally returns to the depot after accomplishing all tasks. During mission execution, the drone can get charged at any recharging station $j \in \mathbb{C}$. The aim of solving the drone delivery problem is to find a route that minimizes the energy cost while being subjected to constraints regarding the delivery mission and drone properties. Further assumptions used in this study are listed as follows:

1) Each customer request is labeled with a predefined load weight.
2) The drone only processes one request at one time.
3) Only fully recharging is considered in this study.
4) The recharge stations could be visited infinite times.
5) The depot also serves as the recharging station during mission execution.

For a better understanding of constraints considered in the DDP-R, the mixed integer programming (MIP) model of the DDP-R with its detailed information and corresponding notation is provided in Appendix A.

B. Drone's Energy Consumption Model with the Wind Effect

For drone delivery, estimating energy requirements performs an essential role in route planning. Unlike ground vehicles, the drone's energy cost can be strongly affected by weather conditions: especially the wind speed and direction because they directly impact the flight kinematic model [32,33]. Nonetheless, most route optimization models only incorporate energy consumption implicitly via a predefined limited drone duration or range. With the aim of better emulating the real-world drone delivery scenarios, we build a stochastic energy consumption model considering varying wind conditions, which is inspired by the work in Refs. [34,35].

1. *Energy Consumption Without Wind*

The total aerodynamic power required for drone flight comprises four parts: parasite power, induced power, profile power, and power required to climb [35]:

$$\begin{aligned} P_{\text{aero}} &= P_{\text{parasite}} + \kappa_{\text{ind}} P_{\text{induced}} + P_{\text{profile}} + P_{\text{gravity}} \\ &= D_{\text{body}} v_a + \kappa_{\text{ind}} T w + \rho A v_T^3 \left(1 + 3\left(\frac{v_a}{v_T}\right)^2\right) \frac{\sigma c_{bd}}{8} \\ &\quad + mg v_a \sin\gamma \end{aligned} \tag{1}$$

where v_a is the airspeed, γ is the flight angle, κ_{ind} is the induced factor (usually equal to 1.15; see Ref. [35]), the body drag force is $D_{\text{body}} = qSC_{D_{\text{body}}} v_a^2$, the thrust is

$$T = \sqrt{m^2 g^2 + D_{\text{drag}}^2 + 2D_{\text{drag}} mg \sin\gamma}$$

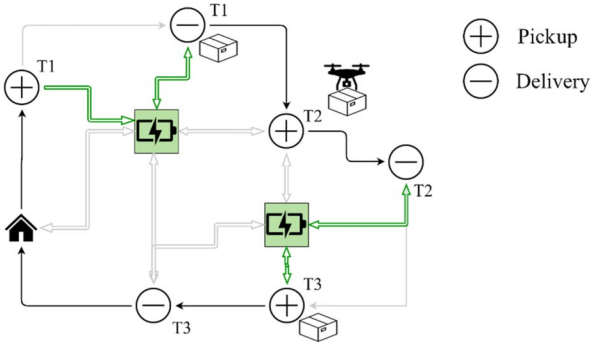

Fig. 1 Drone delivery with recharging.

w is the downwash coefficient, A is the rotor disk area, v_T is the blade tip speed, σ is the rotor solidity ratio, and c_{bd} is the blade drag coefficient. Details about determining w, A, v_T, σ, and c_{bd} are given in Appendix B, along with an overview of notations.

The overall power during flight includes the aerodynamic power P_{aero} (conditioned by the power efficiency η) and the hotel power P_{hotel} (the power required to supply internal electronics) [35]:

$$P = \frac{P_{\text{aero}}}{\eta} + P_{\text{hotel}} \tag{2}$$

For delivery problems, we identify the flight process as having four phases: takeoff, level flight, hovering, and landing. A typical flight profile is shown in Fig. 2. To estimate the total energy demand, the power demand for each phase is weighted with an associated duration:

$$\begin{aligned} E = \frac{1}{\eta} &(t_{\text{tof}} P_{\text{aero}}(m, v_a^{\text{tof}}, \gamma_{\text{tof}}) + t_{lf} P_{\text{aero}}(m, v_a^{lf}, 0 \text{ deg}) \\ &+ t_{\text{hover}} P_{\text{aero}}(m, |v_w|, 0 \text{ deg}) + t_{\text{land}} P_{\text{aero}}(m, v_a^{\text{land}}, \gamma_{\text{land}})) \\ &+ t_{\text{total}} P_{\text{hotel}} \end{aligned} \tag{3}$$

where $|v_w|$ is the wind speed, t_{tof} and t_{land} indicate the times for takeoff and landing, t_{hover} is the time for hovering, t_{lf} is the time for level flight, and t_{total} is the total duration.

In this paper, we consider the drone with vertical takeoff and landing (VTOL), indicating that the ascent angle is $\gamma_{\text{tof}} = 90$ deg and the descent angle $t_{\text{land}} = -90$ deg. Given the altitude of level flight a and the UAV ground speed v_g ($v_g = v_a$ with no wind), the time for level flight is $t_{lf} = d/v_g$, and the time for takeoff and landing is $t_{\text{tof}} = t_{\text{land}} = a/v_v$, where v_v is the vertical speed. Figure 3 depicts the energy consumption curves for a 10 km delivery flight with three different package weights and airspeed varying (left), and its contour map with respect to both airspeed and package weight (right).

2. *Energy Consumption in Presence of Wind*

According to the preceding model, the energy cost for a specific delivery is defined by three variables: package weight m_{pack}, airspeed v_a, and flight time t. Further assuming that the drone is directed by a constant ground speed command, the airspeed is then obtained by integrating the velocity of the wind and the ground speed command, which are derived from the drone kinematic model [36]:

$$\begin{aligned} \dot{x} &= v_g \cos\chi = v_a \cos\psi + v_w \cos\psi_w \\ \dot{z} &= v_g \sin\chi = v_a \sin\psi + v_w \sin\psi_w \end{aligned} \tag{4}$$

where v_g, v_a, and v_w indicate the ground speed, airspeed, and wind speed respectively; $\chi = \tan^{-1}(z/x)$ is the course angle; ψ is the heading angle of the UAV; and ψ_w is the wind course angle. Due to the constant ground speed, the airspeed could be derived as

$$v_a = \sqrt{2v_g^2 + 2v_w^2 - 2v_g v_w \cos(\psi_w - \chi)} \tag{5}$$

In terms of wind modeling, we establish a stochastic wind model in which a constant wind is combined with a random turbulence flow.

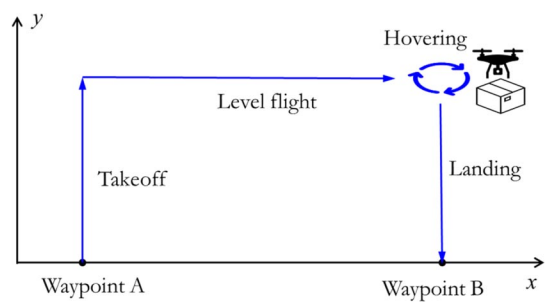

Fig. 2 The flight profile for the VTOL drone delivery, consisting of four phases: takeoff, level flight, hovering, and landing.

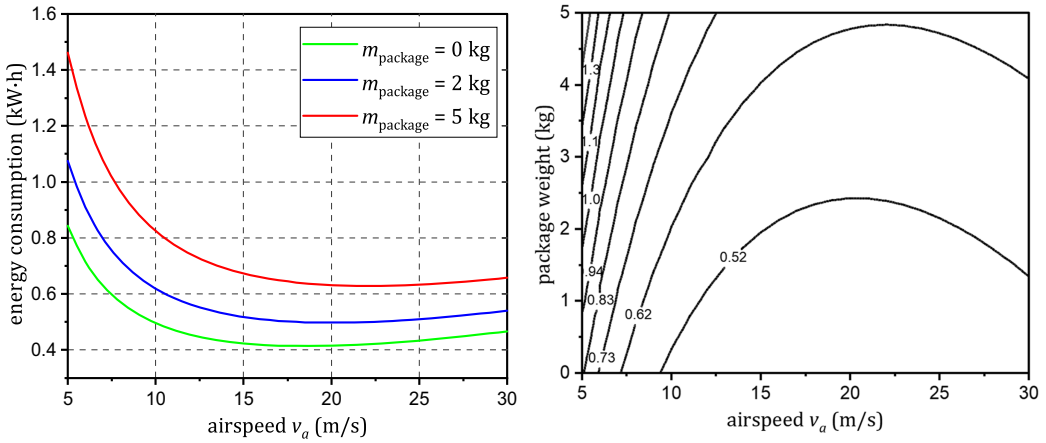

Fig. 3 Energy consumptions for 10 km flight with airspeed and package weight varying: three cases considered with package weights of $m_{\text{load}} = 0, 2$, and 5 kg, respectively (left); and contour map of energy consumption with regard to airspeed and package weight (right). In simulations, no wind is considered, altitude for level flight is $a = 100$ m, hovering time is $t_{\text{hover}} = 5$ s, and vertical speed is assumed as a constant of $v_v = 3$ m/s.

The constant wind speed is randomly generated via a Weibull distribution, which is widely used to describe wind variation [37]. The scale parameter and shape parameter in wind Weibull distributions indicate the mean value of the wind speed and distribution shape, respectively. The direction of the constant wind is assumed to follow a uniform distribution from $\psi_w = 0$ deg to $\psi_w = 360$ deg. The turbulence wind component is generated through the well-known von Kármán turbulence model [38].

To analyze the distribution of energy consumption under varying wind conditions, we conduct a series of Monte Carlo simulations given a oneway waypoint task, where the distance to the target position is 10 km, and the flight is unloaded. After running simulations of 10,000 rounds each with different airspeeds and average wind speed, the statistical figure of the energy consumption can be depicted as in Fig. 4.

The 90% confidence intervals of energy consumption are calculated and depicted as the shaded areas in Fig. 4, where the energy cost value will fall inside with the probability of 90% when the average wind speed is 5 or 10 m/s. As shown from the figure, these two shaded areas are both unignorable. We can also find out that when the airspeed increases, the UAV can resist the wind to some degree, leading to a smaller variance. However, even with a 20 m/s airspeed, winds having an average speed of 5 m/s could also cause a difference in the energy cost up to 0.2 kW·h for a 10 km path segment; and the deviation becomes even larger as wind speed increases. This implies that ignoring variances in energy consumption caused by random wind conditions will degrade the performance of planned routes and even lead to task failures. Meanwhile, in many practical applications, such as drones operating in builtup urban areas or in windy outdoor areas with complex terrain, it is always impossible to obtain accurate wind models. This emphasizes the importance of introducing the stochastic energy consumption model into planning to make the results more wind resistant.

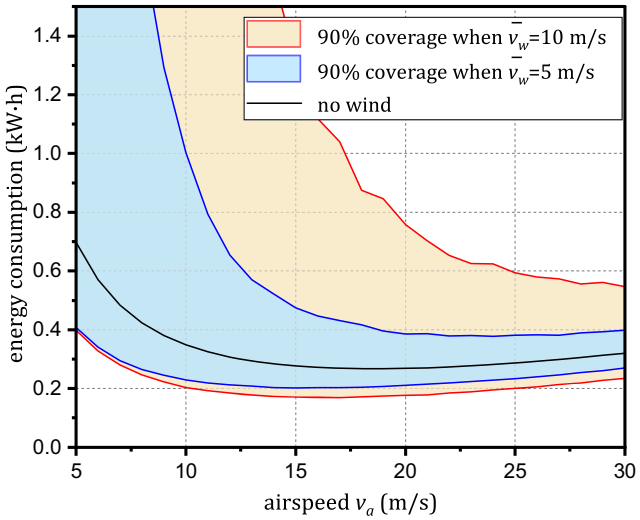

Fig. 4 Statistical distribution of energy consumption for a 10 km flight with the wind. In Monte Carlo simulations, the shape parameter of the wind Weibull distribution is set as $k = 2$.

C. Markov Decision Process for DDP-Rs

For the purpose to handle randomness, we model the stochastic DDP-R as sequential MDPs. This idea, borrowed from path planning algorithms, is naturally well suited for stochastic problems, enabling us to update the route based on the evolution of available information.

A generic MDP consists of four components: state space S, action space A, immediate reward function $r(s, a)$, and state transition probability $p(s'|s, a)$. Specifically, for the DDP-R described earlier in this paper, these components are defined as follows:

1. *State*

The global state $s_i \in S$ is factored into the drone state s_i^D (location, remaining energy) and the mission state s_i^M including locations and status of customer requests, recharging stations, and the depot. The mission state is further decomposed into the vertex features $\boldsymbol{s}^v$ (location) and the edge features $\boldsymbol{s}^e$ (distance, connectivity):

$$\boldsymbol{s} = (\boldsymbol{s}^D, \boldsymbol{s}^M = (\boldsymbol{s}^v, \boldsymbol{s}^e)) \in S \tag{6}$$

where

$$\boldsymbol{s}^D = [x_d, y_d, soc]$$
$$\boldsymbol{s}_j^v = [x_j, y_j, w_j]$$
$$\boldsymbol{s}_{ij}^e = [d_{ij}, a_{ij}]$$

where (x_d, y_d) and (x_j, y_j) is the coordinate of the drone and the vertex j; soc is the state of charge level of the drone; w_j is the weight of the package; d_{ij} is the distance between vertices i and j; and a_{ij} is a binary variable, indicating directional connectivity from vertex i to vertex j.

It is noted that there are two possible route ends:

1) All customer requests have been served.

2) Planning steps reach the maximum step limitation.

When the route comes to an end, the drone is forced to go back to the depot and the state arrives at the goal state $s_f \in S_f$.

2. *Action*

The action indicates the next movement of the UAV, which is represented by the corresponding vertex: $a = j$, $j \in \mathbb{N}$. It could be

the depot, a task, or a recharge station. With the purpose of improving the exploration efficiency, the set of actions is partitioned into valid and invalid sets, and a masking policy is employed to mask invalid actions. The details of the action masking policy will be introduced in Sec. V.C.

3. Reward

The reward function determines the return value obtained from the environment after the system acts. The design of the reward mechanism is critical, which directly guides the training of the RL agent. In the DDP-R studied, our aim is set to minimize the overall energy consumed, which accumulates the energy cost on each path edge. We define an energy cost function ϵ for the drone traversing an edge e, which depends on 1) the payload l, and 2) the wind conditions w. Thus, the reward associated with each travel is set as the negative energy cost of the travel under the configuration (l, w):

$$r(s, a) = -\epsilon(e; l, w) \tag{7}$$

However, with a negative reward function like this, there is an incentive for the UAV to stay at the depot forever to avoid all costs. To encourage the UAV to serve more tasks, an additional negative reward is imposed for tasks left pending when the drone is back at the depot. Specifically, a constant penalty variable C_{pen} is imposed for every task that is left unfinished at the end of the episode, which is denoted as a terminal reward $R^{\mathcal{G}}$:

$$R^{\mathcal{G}}(s_t) = \begin{cases} -C_{\text{pen}} n_{\text{pen}} & \text{if } s_t \in S_f \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where n_{pen} counts the number of pending tasks when the state arrives at s_f.

4. Transition

A transition function describes how states are updated, which is represented by the probability $p(s'|s, a)$ that the current state s is converted to the next state s' by taking the action a. When the UAV travels to the vertex j after choosing action $a = j$, the drone's state and the mission state get updated according to the type of node the drone visited.

a) If the targeted vertex is a task, the status of the drone is updated according to its new location and the energy consumption for the journey by using the model described in Sec. III.B.

b) If the targeted vertex is a recharge station or the depot, the mission status remains unchanged, whereas the location of the drone moves to the targeted vertex and the energy level of the UAV returns to full.

IV. Reinforcement Learning Model

This section describes details about the design of a DRL-based algorithm to solve the problem formulated, including the architecture of the policy network, the masking policy, and the implemented RL training method.

A. Edge-Enhanced Attention Model

The policy network adapts the original attention model (AM) [8] for a better description of the considered DDP-Rs using a *dot-product attention layer with edges*, which is inspired by embedding techniques in the graph neural network. Most policy networks, as discussed in Sec. II, were designed for homovertex problems such as classical TSPs and CVRPs, with only one depot serving as the starting and terminal point. However, in DDP-Rs, there are four different types of nodes (i.e., *pickup tasks, delivery tasks, depot*, and *recharging stations*) coupled with heterogeneous connections subject to the drone's payload limitation and the pickup–delivery precedence constraint. To describe the connectivity status of nodes, we build up an adjacent matrix that together with the distance matrix forms edge features. The adjacent matrix is constituted of binary element e_{ij}, which describes the connectivity from node i to node j. For further elaboration, a toy example is given in Fig. 5, which contains two delivery tasks, one depot, and one charging station. If we define the node order as the depot, stations, pickup tasks, and delivery tasks, its adjacent matrix will be constructed as the binary matrix in Fig. 5. For instance, the first row in the matrix indicates that departing from the depot, the UAV is only expected to pick up parcels or get recharged for long-distance cruise instead of flying back or going to delivery destinations; whereas the first column indicates the vehicle can only return back to the depot from the delivery destinations and recharging stations. Recharging stations are bi-directionally connected with all other nodes, while self-transition is not allowed to avoid repeated charging. For pickup nodes, they are only connected to their corresponding delivery locations and recharging facilities; whereas for delivery tasks, they could direct to any other pickup positions with the empty vehicle, except from its corresponding pickup task because the parcel has just been delivered.

To summarize, the logic behind the connectivity matrix comprises the following:

1) The edge between paired requests is a one-way flight from pickup locations to delivery locations.

2) The drone is not able to load another parcel while loaded.

3) The empty drone is not expected to fly to any delivery destinations.

With the aim to efficiently capture the complex node relationship in DDP-Rs and enhance the network's description ability, the establishment of an adjacent matrix could be regarded as a manual process of feature extraction. To adapt edge features into a policy neural network model, we develop an *edge-enhanced dot-product attention layer* and integrated it into the attention model.

1. Edge-Enhanced Dot-Product Attention Layer

This paper is not the first trial to explore edge information in the graph neural network. Gong and Cheng [28] built a generic framework to sufficiently exploit edge features. The framework can consolidate both graph convolutional networks (GCNs) and graph attention networks (GATs), extending the one-dimensional adjacent matrix to multidimensional edge features. Based on the idea of Ref. [28], Wang et al. [29] and Hussain et al. [39] instantiated the edge augmentation with GATs and the transformer network, respectively.

Although the intention of using extra edge features is to enhance the performance, these edge-integration networks do not always

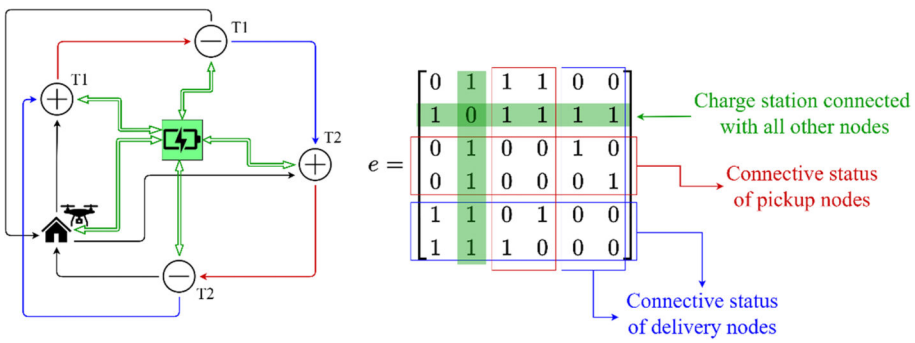

Fig. 5 Example of the adjacent matrix in DDP-Rs. According to delivery logic and payload constraint, the connectivity matrix is defined as shown in the figure.

perform superiorly. Simulations in Ref. [29] revealed that the edge-featured GAT (EGAT) even has a slight performance degradation as compared to the GAT when executing node-sensitive tasks although it is achieving higher accuracy on edge-sensitive tasks (e.g., trading network classifications). As analyzed in Ref. [29], this is due to the fact that in the EGAT, edge features are updated by integrating its adjacent edges. These features, however, are most likely to be useless in node-sensitive tasks, and interferences may occur because of the intensive updating. To avoid the interference from other adjacent edges, we adopt another architecture of edge integration for the concerned routing problem, which is identified as node sensitive. The difference is illustrated in Fig. 6. In the EGAT model proposed by Ref. [29], both of the node features H and the edge features E get evolved by each EGAT layer, whereas the proposed EGAT only updates the node features retaining edge features that are the same for each layer. To implement the model, we propose an edge-enhanced dot-product attention layer, which facilitates the network model to use edge features without updating it frequently.

The attention mechanism performs a message passing process over the nodes of a graph: during which, weights are added when integrating neighborhood elements. In the proposed edge-enhanced attention layer, edge features are taken into account when calculating weights and merging values. Specifically, an attention layer originally integrates the *value* of the node's neighbors v_i, and weights them using the *compatibility* of its *query* q_i with the *key* of the neighbor k_i. To take advantage of edge information, we integrate the embedded edge feature into the compatibility and merge it into the value of the neighbors, as illustrated in Fig. 7.

Formally, the key, value, and query for each node are computed by projecting node embedding h_i:

$$q_i = W^Q h_i \quad k_i = W^K h_i \quad v_i = W^V h_i \tag{9}$$

where $W^Q, W^K \in \mathbb{R}^{d_k \times d_h}$ and $W^V \in \mathbb{R}^{d_v \times d_h}$, d_k and d_v are designable dimensions.

In addition to node projection, edge embedding e_{ij} is linearly projected, constructing the edge component for the compatibility and the value of each edge:

$$c^e_{ij} = W^{C_e} e_{ij} \quad v^e_{ij} = W^{V_e} e_{ij} \tag{10}$$

Here, learnable parameter $W^{C_e} \in \mathbb{R}^{1 \times d_h}$ is used to generate a one-dimensional *edge compatibility* matrix c^e_{ij}, and $W^{V_e} \in \mathbb{R}^{d_v \times d_h}$ has the same dimensions as the node weight W^V.

The compatibility is then calculated as the sum of its edge component and the dot product of the query from node i and the *key* from node j. From the compatibilities, we compute the attention weights $a_{ij} \in [0, 1]$ using a softmax function:

$$a_{ij} = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}} \tag{11}$$

Finally, the vector h'_i that is received by node i is the weighted sum of the *node value* v_i and the *edge value* v^e_{ij}:

$$h'_i = \sum_j a_{ij}(v_j + v^e_{ij}) \tag{12}$$

To enhance the capability of expression and benefit the stability of the attention learning process, the single-layer attention is then extended to the *multihead attention network*. Specifically, we compute M attention layers with independent parameters, using $d_k = d_v = d_h/M$, and then concatenate their outputs to the single d_h-dimensional vector. The final multihead attention value for the node is

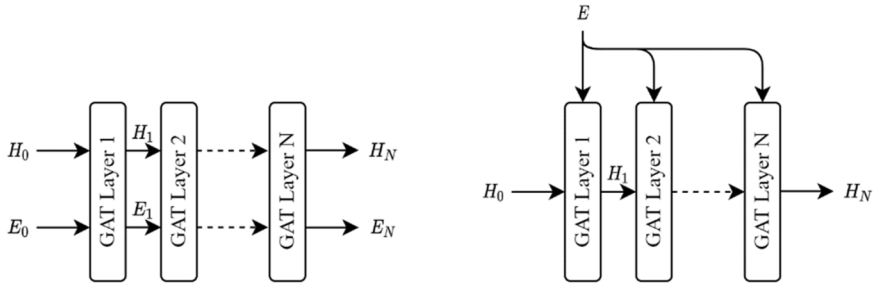

Fig. 6 Two edge-enhanced GAT architectures. In the first architecture, both node features and edge features are evolved for each attention layer (left), whereas the second architecture only updates the node features (right).

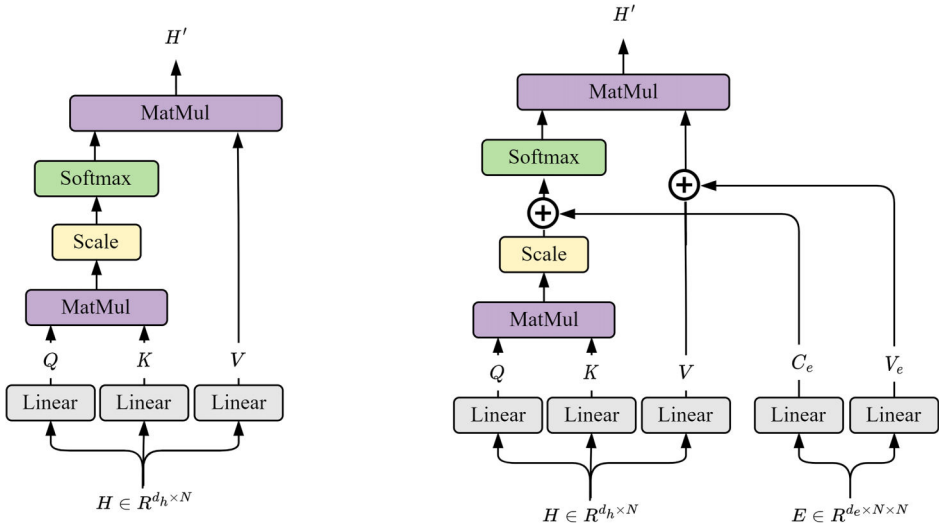

Fig. 7 Scaled dot-product attention (left), and edge-enhanced scaled dot-product attention (right). In edge-enhanced scaled dot-product attention, edge features are projected and then merged into original value vector and compatibility vector, i.e., the scaled dot-product of query and key vectors (MatMul = matrix multiplication).

$$h_i' = \|_{m=1}^{M} \sum_j a_{ij}^{(m)} \left(v_j^{(m)} + v_{ij}^{e(m)} \right) \tag{13}$$

where $\|$ represents concatenation, and the superscript (m) indicates parameters obtained by the mth attention mechanism.

2. Encoder–Decoder Architecture

Figure 8 depicts the overall architecture of our policy network, called the attention model with edges. The AM-E inherits the encoder–decoder structure of the transformer model, which is recognized as the most competitive neural sequence transduction model [40]. Taking a toy instance (two delivery requests, one recharging station, and one depot), for example, the vertex features $[\, s_0^v \;\; \cdots \;\; s_5^v \,]$ and edge features s^e are first passed to the attention module which is denoted as the dashed-line block in the figure after being initialized. Then, the encoded features are recursively updated by the attention module N times until we get the output of the encoder: the embedded node features $[\, h_0^N \;\; \cdots \;\; h_5^N \,]$. Afterward, using drone features s^D as the query vector and masking unavailable nodes, we can finally get the policy output $[\, p_0 \;\; \cdots \;\; p_5 \,]$, which is the policy probability of each node.

Regarding the encoder, a similar structure to the transformer [40] is adopted, which provides a mapping from the node raw features to a richer embedding space, through which the node's own features, the features of its neighbors, and the features of the edge connected with neighbors are all represented. Specifically, mission features $[s^v, s^e]$ are initially embedded to a larger dimension d_h via the nodewise linear projection. The projection parameters are only shared among features of the same category (pickup nodes, delivery nodes, recharging nodes, and the depot). Then, the embeddings $h_i^{(0)}$ are recursively updated by N multihead attention layers, with each consisting of three operations: multihead attention, feed forward, and normalization (see Ref. [8] for details on the rest of the encoder).

The decoder of the policy network leverages the node embedding from the encoder and generates a probability vector $\boldsymbol{p}$ for selecting nodes at each step. To achieve this, the graph is augmented with a special *context node* (c) to represent the decoding context. Herein, the context of the DDP-R consists of the embedding of the graph and the current state of the drone:

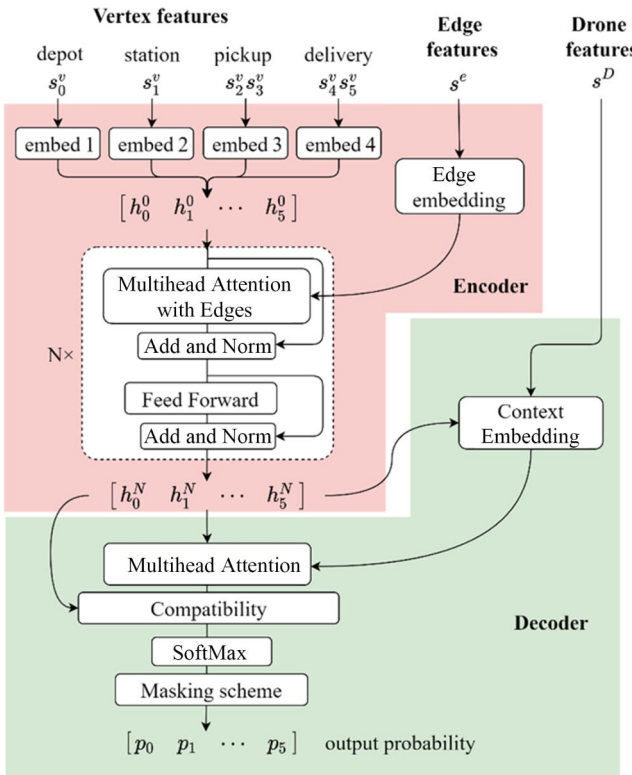

Fig. 8 AM-E policy network.

$$h_{(c)}^{(N)} = \left[h_{(g)}^{(N)}, h_{(d)} \right] \tag{14}$$

where the context $h_{(c)}$ is the concatenation of an aggregated node embedding $h_{(g)}^{(N)}$:

$$h_{(g)}^{(N)} = \frac{1}{n} \sum_{i=1}^{n} h_i^{(N)} \tag{15}$$

and $h_{(d)}$ is the linear projection of the drone state s^D:

$$h_{(d)} = W^{(d)} s^D + b^{(d)} \tag{16}$$

Then, the context embedding is computed using a multihead attention mechanism, with a single query $q_{(c)}$ from the context node as well as the keys and values from the node embedding of the encoder:

$$h_{(c)}^{(N+1)} = \text{multihead}(W_c^Q h_{(c)}, W_c^K h^{(N)}, W_c^V h^{(N)}) \tag{17}$$

Given the context embedding output $h_{(c)}^{(N+1)}$, we add one final attention layer to compute the output probabilities, for which we only compute the compatibilities with $q_{(c)} = W_c^Q h_{(c)}^{(N+1)}$ and $k_i = W_c^K h_i^{(N)}$. The compatibility is then clipped within $[-C, C]$ using tanh, which is followed by the masking policy described in Sec. IV.C:

$$u_{(c)j} = \begin{cases} C \cdot \tanh\left(\frac{q_{(c)}^T k_j}{\sqrt{d_k}} \right) & \text{if } j \text{ is valid} \\ -\infty & \text{otherwise} \end{cases} \tag{18}$$

Finally, after a softmax function, each node is scored with a probability as the final output, which given as

$$p_i = p_\theta(\pi_i | s) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}} \tag{19}$$

B. Masking Scheme

With the purpose of improving the exploration efficiency and ensuring the feasibility of solutions, the set of actions is masked by a designed masking scheme to exclude infeasible routes. According to the battery capacity constraint and the pickup–delivery precedence constraint, an action of $a = v_j$ is labeled as valid only if all of the following conditions are satisfied:

1) If v_j represents a pickup or delivery task, the task must not have been served, and the UAV has enough energy to cover the trip to the vertex v_j and return to the closest charging station.

2) If v_j represents a charging station (or the depot), it should be reachable from the vertex where the UAV is currently located within its remaining battery capacity.

3) If the previous node visited is a pickup task, all other nodes except charging stations, the depot, and the corresponding delivery task are masked.

4) If the previous node visited is a delivery task, or if the UAV just departs from the depot, all delivery tasks will be masked.

5) If the last node visited is a charging station (or the depot), all station nodes and the depot node will be masked. Regarding task nodes, it depends on what type of node was visited before the charging station (or the depot). Then, execute masking according to point 3 and point 4.

It is noted that even for stochastic cases in the presence of winds, the energy amount for paths is still calculated using a nonwind energy prediction model because the wind information is assumed to be unknown before planning.

C. Training Method

Shown as Algorithm 1, we train the policy network using the reinforce algorithm with baselines. The baseline is chosen as a rollout or critic according to whether the problem is deterministic or

Algorithm 1: Reinforce learning algorithm

Initialize: policy parameter θ, rollout baseline parameter ϕ, and paired t test threshold α if *rollout baseline*: critic baseline parameter θ_v if *critic baseline*
Input: number of epochs N, batch size B
for each training $epoch = 1, 2, \ldots, N$, **do**
  generate instances of batch size B;
  for each instance $b = 1, 2, \ldots, B$, **do /* ran in parallel */**
    sample trajectory using policy π_θ;
    receive the accumulative reward of the trajectory R^b;
    if rollout baseline, **then**
      receive baseline reward $v(s_0^b)$ using GreedyRollout policy π_ϕ;
    else, if critic baseline, **then**
      estimate values of initial states using critic $v(s_0^b)$;
    end
  end
  estimate mean policy gradient on the batch of trajectories:
  $d\theta = \frac{1}{B}\sum_{b=1}^{B}(R^b - v(s_0^b))\nabla_\theta \log \pi_\theta(\pi_b|s_0)$;
  if rollout baseline, **then**
    update policy parameters θ;
    if $OneSidedPairedTest(\pi_\theta, \pi_\phi) < \alpha$:
      replace ϕ using θ;
    End
  else, if critic baseline, **then**
    estimate the mean gradient of the mean squared error of the critic on the batch of trajectories:
    $d\theta_v = \frac{1}{B}\sum_{b=1}^{B}\partial(R^b - v(s_0^b))^2/\partial\theta_v$;
    update policy parameters θ and critic parameters θ_v;
  end
end

stochastic. In detail, at the beginning of each episode, one batch of new samples is stochastically generated. Then, the routes are sequentially sampled according to the probability output from the policy network, and rewards are collected. Moreover, we get the expected reward $v(s_0^b)$ from the critic network or the rollout baseline network, which is used to calculate the advantage component of the gradients. Backpropagation is then adopted to update the policy network and the critic network for the critic baseline. Regarding the use of a rollout baseline, at the end of each episode, the parameter of the baseline policy network will be replaced by that of the policy network if the performance of the latter is significantly superior.

V. Numerical Simulations

To verify the effectiveness and evaluate the performance of the proposed AM-E, we conduct simulations for solving DDP-Rs in the presence and absence of winds. In this section, we will introduce our experimental setup, and investigate the performance of the proposed method in comparison to the original AM and other state-of-the-art solutions.

All experiments are carried out on a 16-cores Intel E5-2620 v4 CPU, a Tesla K80 GPU, or a Tesla V100 GPU.

A. Experimental Setup

1. *Simulated Environment*

A simulated environment is built to imitate missions of the DDP-R, from which we can sample trajectories and get rewards back to train networks or evaluate the planning models. For initialization, the pickup tasks, delivery tasks, charging stations, and depot are randomly located on a 10×10 km mission area. The distance between every two nodes is set as the Euclidean distance. In simulations, the airspeed of the UAV is set to a constant 20 m/s and a battery capacity of 1.5 kW · h is assumed. The energy cost on each edge is decided by the energy consumption model presented in Sec. III.B, which is supposed to be deterministic if no wind or stochastic with the wind. Parameters of the constant wind component (i.e., direction and speed) are generated following the distribution model in Sec. III.B and assumed to remain unchanged for each path segment. Simulation environments are designed with three flexible dimensions: number of tasks, number of recharging stations, and deterministic/stochastic operating environment. For ease of clarification, we will include these information in the problem name in the following content. For example, "DDP10-R3" indicates the deterministic drone delivery problem of 10 tasks with 3 recharging stations, and "S-DDP40-R3" indicates the stochastic drone delivery problem of 40 tasks with 3 recharging stations.

2. *Network Structure*

For the proposed AM-E, the node embedding, edge embedding, and context embedding are all one-layer elementwise liner projections with 128 dimensions. The multihead graph attention network consists of $M = 8$ heads computing key vectors and value vectors of dimension $d_k = d_v = 16$. The number of sequential multiattention modules is three. In the feedforward layer, the node features are passed through nodewise projections of one hidden sublayer with the Rectified Linear Unit (ReLU) activation and 512 hidden units.

3. *Training Parameters*

We adopt the Adam optimizer to train the network with a constant learning rate of $\alpha = 10^{-4}$. We run training for 100 epochs for each problem. In one epoch, we process 1.28 million instances in 2500 iterations with a batch size of 512. Each epoch takes around 15 min for training DDP20-R3 using a Tesla K80 GPU, 42 min for DDP40-R3 using a Tesla K80 GPU, and 28 min for DDP80-R3 using a Tesla V100 GPU.

B. Validation of Edge Feature Enhancement

To test the efficacy of edge enhancement for solving DDP-Rs, we carried out a comparison study among three neural network models: the proposed AM-E, the original attention model [8], and the heterogeneous attention model [27]. As mentioned, the heterogeneous attention model is designed for pickup and delivery problems featuring a better performance by considering heterogeneous roles in PDPs. It does not exactly match the DDP-R studied in this paper because of the presence of recharging stations, and so an extension is made that two more types of attention layers are added regarding pickup–recharge relations and delivery–recharge relations, respectively.

The learning curves of the preceding three models are depicted in Fig. 9. The total number of trainable parameters and the training time for one episode are summarized in Table 1. Simulation results in terms of the basic travel sales problems and the concerned DDP-Rs are shown in Table 2.

From Table 2, in the case that all settings are identical except for model architectures, the AM-E and heterogeneous attention model show superiority over the AM in solving DDP-Rs, whereas no

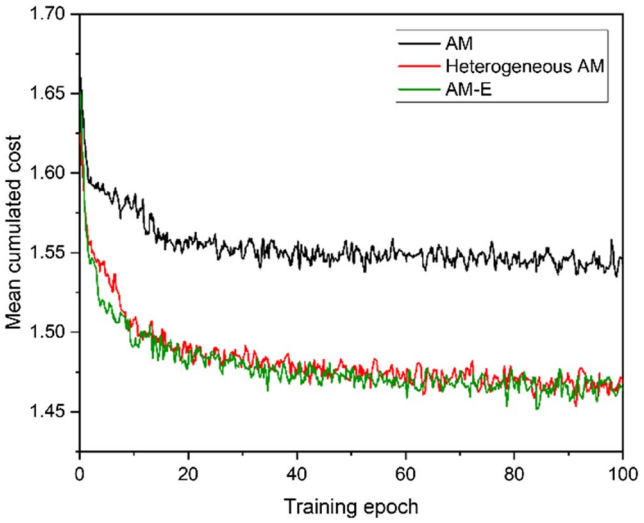

Fig. 9 Learning curves for solving the DDP-R with 10 delivery requests and three recharge stations.

Table 1 Total trainable parameters of models and running time for one epoch[a]

	AM	Heterogeneous AM	AM-E
Trainable parameters	693,632	1,086,848	746,112
Running time	4 min, 50 s	8 min, 18 s	6 min, 7 s

[a]All programs ran in Tesla V100, and time was measured over the entire training set and averaged.

Table 2 Comparison among AM, AM-E, and heterogeneous AM[a]

Task	Optimal	AM	HeterogeneousAM	AM-E
TSP20	3.83[a]	3.84 (0.33%)[a]	3.84 (0.33%)[a]	3.84 (0.33%)
TSP50	5.69[a]	5.82 (2.28%)[a]	5.82 (2.28%)[a]	5.81 (2.11%)
TSP100	7.76[a]	8.19 (5.49%)[a]	8.19 (5.49%)[a]	8.08 (4.12%)
DDP10-R3	1.483	1.590 (7.36%)	1.527 (2.97%)	1.526 (2.90%)
DDP20-R3	2.688	2.864 (6.55%)	2.788 (3.72%)	2.790 (3.79%)
DDP40-R3	— —	5.286 (2.66%)	5.170 (0.41%)	5.149 (0.00%)

[a]Results reported in Ref. [8].

significant difference is shown for TSPs with homogeneous and fully connected nodes. It should be noted that the heterogeneous attention model degenerates to the AM in solving the TSP, and so we refer to the same results in the table.

For further comparison, the proposed AM-E achieves comparable and even slightly better performance than the heterogeneous attention model with about a 32% reduction in parameters. In the heterogeneous attention model, eight extra types of attention layers are added to the original one for solving DDP-Rs. Even though the parameters of all keys and values are shared, the query matrices of the attention layers are kept independently, which introduce a large number of extra trainable parameters to the model and almost double the running time for one episode. More trainable parameters always imply higher computational cost and a larger memory requirement. In the proposed AM-E, we only use one extra edge matrix to capture the node relationships, which requires much less trainable parameters and avoids complex network formulation while retaining comparable performance.

C. DDP-Rs Without Winds

In the first experiment, we test the performance of the proposed AM-E for solving the deterministic DDP-R without considering the wind. The experiment is carried out with three different dimensions (the numbers of delivery requests are $N = 10$, 20, and 40; whereas the number of charging stations is fixed to three) and compared to the original AM and deterministic baseline methods.

The first baseline in comparison is obtained by Gurobi, which is a mathematical optimization solver, to provide the exact optimal solution for evaluation. To avoid an unaffordable computing time, we set a 100 s time limit for Gurobi. Under these limitations, Gurobi sometimes fails to obtain a feasible solution, and so we count the success rate for the Gurobi baseline. The second baseline comes from Google's OR-Tools backed by the CP-SAT solver, which is usually seen as the most advanced heuristic solver. We gather OR-Tools solutions of different qualities using the 1, 10, and 50 s solving times, respectively.

Regarding the DRL implementations, we introduce the original AM as another baseline. For the heterogeneous AM, because the proposed AM-E outperforms it in terms of both solution qualities and computing efficiencies (as demonstrated by simulations in the last section), we remove the heterogeneous AM from the following comparison studies. With regard to the proposed AM-E and the baseline AM, we apply two decoding strategies for evaluation:

1) The first decoding strategy is *greedy*, for which we always select the action with maximum probability at each step.

2) The second decoding strategy is *sampling*, for which we sample $N = 1280$ routes for each instance according to the probability distribution and choose the one with minimum cost.

The simulation results are summarized in Table 3.

From Table 3, Gurobi fails to obtain a feasible solution in some instances within 100 s, and the proportion of these cases becomes larger as the problem size increases. OR-Tools obtains high-quality solutions to small-size problems but scales poorly because the time required for searching a high-quality route explodes as the number of delivery requests increases. Comparing these two DRL-based methods, the proposed AM-E shows a performance improvement over its basic form, which is the AM, both for greedy decoding and sampling decoding. Overall, the AM-E has better scalability, and it efficiently produces high-quality routes for all three dimensions.

D. DDP-Rs in Presence of Winds

Next, the robustness of the proposed model is tested against the stochastic edge cost in DDP-Rs under varying wind conditions. Regarding stochastic energy costs, if the drone's onboard battery is used up before it gets recharged, a penalty of $p = 1.0$ will be imposed on the overall cost. The model is trained in a stochastic environment with the mean wind value of $\bar{v}_w = 5$ m/s, and then it is tested for robustness with $\bar{v}_w = 2$, 5, and 10 m/s. Because the DRL agent performs as an online policy, it is expected to continuously adjust the route according to its current battery level. However, the Gurobi and OR-Tools, as offline planning methods, use predefined routes with less flexibility. Thus, in order to prevent battery exhaustion, a margin of safety is implemented to these baselines; i.e., the amount of available battery capacity used in planning is reduced by a specific percentage.

To ascertain the best value of the margin of safety, we obtain expected costs and actual costs via OR-Tools using different margins for solving S-DDP10-R3 and S-DDP40-R3, presented in Fig. 10. As seen in Fig. 10, small margins turn out to be risky, resulting in large biases between expected and actual costs; whereas overly large margins are conservative, resulting in higher costs for frequent charging station visits. A margin of safety of around 20–30% provides the best planning results with the lowest costs in actual executions.

Table 4 compares the proposed AM-E, AM, and offline planning baselines with the 20 and 30% margins of safety. It is noted that the results of Gurobi are excluded from the comparison for solving S-DDP40-R3 due to its high failure rate. Like deterministic cases, OR-Tools gains solutions of lower costs for small-size deliveries, whereas DRL methods perform better with scenarios of 20 requests

Table 3 Costs and computing times for solving deterministic DDP-Rs

	DDP10-R3			DDP20-R3			DDP40-R3		
Method	Cost	Gap, %	Time, s	Cost	Gap, %	Time, s	Cost	Gap, %	Time, s
Gurobi (optimal)	1.483 ± 0.004(99.88%)	0.13	2.436	2.688 ± 0.016(98.90%)	0.00	4.306	4.925 ± 0.380(88.00%)	— —	46.064
OR-Tools (1 s)	1.493 ± 0.036	0.81	1.0	3.438 ± 0.034	27.90	1.0	— —	— —	— —
OR-Tools (10 s)	1.481 ± 0.036	0.00	10.0	2.752 ± 0.005	2.38	10.0	10.007 ± 6.172	98.59	10.0
OR-Tools (50 s)	— —	— —	— —	2.726 ± 0.261	1.41	50.0	6.452 ± 0.068	28.04	50.0
AM (greedy)	1.590 ± 0.004	7.36	0.109	2.864 ± 0.006	6.55	0.181	5.286 ± 0.008	4.90	0.280
AM (sample 1280)	1.521 ± 0.004	2.70	0.141	2.741 ± 0.005	1.97	0.251	5.094 ± 0.007	1.09	0.380
AM-E (greedy)	1.530 ± 0.005	3.04	0.063	2.776 ± 0.006	3.79	0.121	5.131 ± 0.007	2.18	0.416
AM-E (sample 1280)	1.487 ± 0.004	0.41	0.131	2.709 ± 0.005	0.79	0.286	5.039 ± 0.007	0.00	0.465

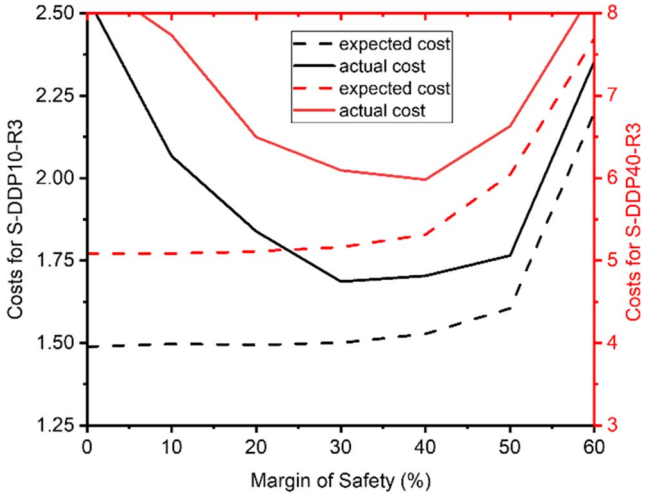

Fig. 10 Expected costs and actual costs with different margins of safety.

or 40 requests. Setting certain margins for battery usage is the strategy mostly used in industrial applications, and it successfully guarantees a certain degree of robustness for these deterministic baselines, as shown in the Table 4. However, how to define the margin value while operating in varying wind fields is a tricky question. It might become either risky or conservative. As an alternative solution, the DRL approaches avoid this problem by continuously adjusting the route according to the UAV's remaining energy.

Concerning the comparison with varying mean wind values, OR-Tools obtains the best results with the wind of $\bar{v}_w = 2$ m/s, whereas the AM-E shows better robustness when planning with windier scenarios. We also notice that OR-Tools, as an offline planner, causes large variances when executing with $\bar{v}_w = 10$ m/s, indicating that the drone fails to return to stations for some instances. At the same time, even though the AM-E policy was also trained with a milder wind field, it remains a stable performance demonstrated by the smaller variances in presence of a larger wind magnitude of $\bar{v}_w = 10$ m/s, which also implies that the RL method have a certain level of transferring ability.

Overall, the proposed method obtains high-quality routes and arrives at the solutions in a very short time. The improvement becomes more significant when handling larger scales, demonstrating the AM-E with better scalability than Gurobi or OR-Tools, as also seen by its stable performance and quick processing times. Again, the AM-E offers better solutions than the AM without consuming more time. Moreover, because the DRL-based method possesses the ability to respond to environmental changes, it is expected to perform better in dynamic scenarios as compared to the deterministic approaches that require replanning of whole routes.

E. Generalization with Dimensions

In all of the aforementioned simulations, the AM-E was trained and tested separately on each problem dimension such that model weights were specialized on these dimensions. Despite the fact that a route planner could transfer across these models according to current task requirements, we are still expecting a general model that performs well in all cases to facilitate the future implementations. To that end, the proposed model has been designed in a dimension-independent way, decorated with a nodewise embedding layer and a mean-reduction value layer for the critic baseline. Models are able to solve problems of other dimensions without the parameter structure inconsistency.

Figure 11 and Table 5 record the intersective validation results of models for solving delivery problems with $N = 10$, $N = 20$, and $N = 40$, where the bottom axis in the figure refers to the problem size the model learned, and the top axis refers to the problems to be solved. We observed that although the corresponding model achieves the smallest cost in comparison with those learned for the other size, the differences across them are very small for all dimensions. More interestingly, we notice that even though the policy for DDP10 was only trained with problems of small sizes, its degradation in solving DDP40 is still acceptable, which is quite meaningful in real-life situations because generalizing to larger problem sizes can save tremendous time and computing resources for large applications.

Table 4 Costs and computing time for solving stochastic DDP-Rs

		S-DDP10-R3		S-DDP20-R3		S-DDP40-R3	
$\bar{v}_w$, m/s	Method	Cost	Gap, %	Cost	Gap, %	Cost	Gap, %
5	Gurobi (mgn. 20%)	1.783 ± 0.816(99.90%)	5.69	3.161 ± 1.185(97.80%)	1.38	6.423 ± 2.674(41%)	— —
	Gurobi (mgn. 30%)	1.716 ± 0.600(99.89%)	1.72	3.145 ± 1.138(98.30%)	0.87	6.209 ± 2.418(52%)	— —
	OR-Tools (mgn. 20%)	1.839 ± 0.672	9.01	3.418 ± 1.581	9.62	6.495 ± 3.431	11.98
	OR-Tools (mgn. 30%)	1.687 ± 0.300	0.00	3.341 ± 1.255	7.15	6.097 ± 3.045	5.12
	AM (greedy)	1.781 ± 0.007	5.57	3.234 ± 0.012	3.72	5.937 ± 0.019	2.36
	AM-E (greedy)	1.744 ± 0.009	3.38	3.118 ± 0.011	0.00	5.800 ± 0.019	0.00
2	OR-Tools (mgn. 20%)	1.528 ± 0.211	0.00	2.806 ± 0.282	0.00	5.396 ± 1.089	0.00
	OR-Tools (mgn. 30%)	1.544 ± 0.262	1.05	2.822 ± 0.327	0.57	5.782 ± 2.065	7.15
	AM (greedy)	1.657 ± 0.017	8.44	2.962 ± 0.021	5.56	5.511 ± 0.033	2.13
	AM-E (greedy)	1.584 ± 0.013	3.66	2.880 ± 0.019	2.64	5.458 ± 0.033	1.15
10	OR-Tools (mgn. 20%)	3.485 ± 2.237	64.54	6.482 ± 3.878	61.40	13.690 ± 8.028	71.17
	OR-Tools (mgn. 30%)	2.874 ± 1.653	35.69	5.160 ± 2.648	28.48	11.038 ± 5.666	38.01
	AM (greedy)	2.379 ± 0.051	12.32	4.216 ± 0.076	4.98	8.005 ± 0.137	0.08
	AM-E (greedy)	2.118 ± 0.032	0.00	4.016 ± 0.068	0.00	7.998 ± 0.143	0.00

mgn. = battery safety margin.

Table 5 Costs and computing times for solving stochastic DDP-Rs

	DDP10-R3		DDP20-R3		DDP40-R3	
Method	Cost	Gap, %	Cost	Gap, %	Cost	Gap, %
AM-E (greedy) (DDP10)	1.530 ± 0.015	0.00	2.816 ± 0.022	1.44	5.266 ± 0.027	2.63
AM-E (greedy) (DDP20)	1.532 ± 0.013	0.13	2.776 ± 0.017	0.00	5.147 ± 0.023	0.31
AM-E (greedy) (DDP40)	1.544 ± 0.013	0.92	2.783 ± 0.017	0.25	5.131 ± 0.023	0.00

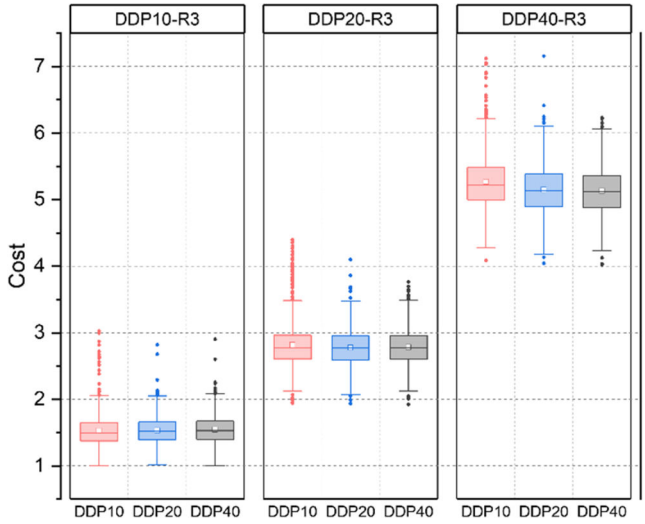

Fig. 11 Generalization performance of AM-E.

VI. Future Work

The extension to multidrone cases is desired by most real-world applications. Given that the ability of the single drone is limited (and with advances in autonomous technologies), many realistic applications prefer the collaboration of multiple UAVs, which tend to be more efficient and flexible. Future research opportunities involve the extension of the current work to the delivery problem with multiple UAVs, where a decentralized planning algorithm integrated with interdrone communication will mainly be investigated.

In addition, the training dataset used in this work is entirely produced numerically. Generalizing the trained network to the real logistic application remains a concern. It is planned to collect more realistic data from UAV delivery operations for validation. Building a more refined simulation environment that considers time windows, the quality of service, and dynamic events is included in future research plans.

VII. Conclusions

In this study, a new variant of the vehicle routing problem has been investigated, which is proposed to target the drone application for parcel delivery. The routing model includes recharging operations and considers the effect of varying winds. Specifically, a simulated environment was built with a stochastic energy consumption model under varying winds to imitate real-world delivery scenarios in which the energy-constrained UAV could be recharged in the middle of mission execution. To address the delivery problem efficiently, a DRL-based method with an AM-E network has been presented and demonstrated with high-quality solutions. The AM-E is composed of a novel edge-enhanced dot-product attention layer via merging edge features into the information propagation, offering the network a stronger ability to describe the complex relationship among heterogeneous nodes. With the designed masking policy and reward function, the AM-E has been trained and evaluated via the simulated environment. The results showed the AM-E is fast, robust, and has better scalability as compared to three baseline heuristic methods and a state-of-the-art deep learning model.

Appendix A: Mixed Integer Programming Model for DDP-Rs

With the objective to minimize the energy cost, the concerned DDP-Rs claimed in Sec. III.A are formulated into mixed integer programming. The indices, sets, parameters, and decision variables in the MIP model are defined in Table A1 and the following:

The proposed MIP model is as follows:

$$\min \quad J = \sum_{i,j \in N} E_{ij} x_{ij}$$

Table A1 Notations used in the MIP model

Variable	Description
	Indices and sets
i, j	Indices of nodes
$\mathbb{P}/\mathbb{D}$	Set of pickup/delivery nodes, $\mathbb{P} = \{1, 2, \ldots, n\}$, $\mathbb{D} = \{n+1, n+2, \ldots, 2n\}$
$\mathbb{C}$	Set of recharging stations
$\mathbb{O}$	Set of the depot node and its one copy, $\mathbb{O} = \{n_0, n_0'\}$
$\mathbb{N}$	Set of all nodes, $\mathbb{N} = \mathbb{O} \cup \mathbb{P} \cup \mathbb{D} \cup \mathbb{C}$
	Decision parameters
x_{ij}	One if vehicle travels from node i to node j; zero otherwise
e_i, %	State of charge (SOC) when leaving node i
t_{ij}	Vehicle departure time at node i if the vehicle comes from node i
	Parameters
n_0	Node for the UAV depot
n_0'	Node for the copy of the UAV depot
E_{ij}, kW·h	Electric energy needed to travel from node i to node j
$e_{\min}$, %	Lower bound of SOC constraints
$e_{\max}$, %	Upper bound of SOC constraints
$E_{\max}$, kW·h	Maximum battery capacity
τ_{ij}	Travel time from node i to node j
M	A big positive value

subject to

$$x_{ii} = 0, \ \forall i \in \mathbb{N}$$

$$\sum_{i \in \mathbb{N}} x_{i,n_0} = 0$$

$$\sum_{j \in \mathbb{N}} x_{n_0',j} = 0$$

$$\sum_{j \in \mathbb{N}} x_{ij} = 1, \ \forall i \in \mathbb{P} \cup \mathbb{D} \cup \{n_0\}$$

$$\sum_{i \in \mathbb{N}} x_{ij} = 1, \ \forall j \in \mathbb{P} \cup \mathbb{D} \cup \{n_0'\}$$

$$\sum_{i \in \mathbb{N}} x_{ij} = \sum_{i \in \mathbb{N}} x_{ji}, \ \forall j \in \mathbb{C}$$

$$-M(1-x_{ij}) \le x_{j,n+i} - 1 \le M(1-x_{ij}), \ \forall i \in \mathbb{P}, \ \forall j \in \mathbb{C}$$

$$-M(x_{i,n+i}) \le \sum_{j \in \mathbb{C}} x_{ij} - 1 \le M(x_{i,n+i}), \ \forall i \in \mathbb{P}$$

$$-M(1-x_{ij}) \le e_i - e_j - \frac{E_{ij}}{E_{\max}} \le M(1-x_{ij}), \ \forall i \in \mathbb{P} \cup \mathbb{D}, \ \forall j \in \mathbb{N}$$

$$-M(1-x_{ij}) \le e_{\max} - e_j - \frac{E_{ij}}{E_{\max}} \le M(1-x_{ij}), \ \forall i \in \mathbb{C} \cup \{n_0\}, \ \forall j \in \mathbb{N}$$

$$e_{n_0} = e_{\max}$$

$$e_{\min} \le e_i \le e_{\max}, \ \forall i \in \mathbb{N}$$

$$t_{n_0} = 0, \ \forall j \in \mathbb{N}$$

$$-M(1-x_{ij}) \le t_i + \tau_{ij} - t_j \le M(1-x_{ij}), \ \forall i \in \mathbb{N}, \ \forall j \in \mathbb{N}/\mathbb{C}$$

$$x_{ij} \in \{0,1\}, \ \forall i, j \in \mathbb{N}$$

Appendix B: Energy Consumption Model

Let n_{rotor} denote the number of rotors, n_{blade} the number of blades per rotor, and r the radius of the rotors. The total area on which air is moved by the rotors R is calculated as

$$R = r^2 \pi n_{\text{rotor}}$$

The speed of the blade tips v_t depends on the thrust to be exerted and the physical property of the blades. Therefore, letting $\bar{c}$ denote the

Table B1 Parameter values used in the drone energy consumption model

Term	Symbol	Value
Tare weight, kg	m_{tare}	10.886
Air density, kg/m^3	ρ	1.225
Acceleration of gravity, m/s^2	g	9.807
Frontal surface area, m^2	A	0.15
Hotel power during flight, kW	P_{hotel}	0.1
Battery capacity, kW · h	C_{batt}	1.5
Engine efficiency	η	0.9
Number of rotors	n_{rotor}	8
Number of blades	n_{blade}	3
Rotor radius, m	r	0.4
Air drag	$C_{D_{\text{body}}}$	0.3
Blade drag	c_{bd}	0.075
Rotor mean chord	$\bar{c}$	0.1
Blade lift	$\bar{c}_l$	0.4
Lifting power markup	κ_{ind}	1.15

rotor mean chord and $\bar{c}_l$ the mean lift coefficient, the blade speed follows as

$$v_t = \sqrt{\frac{6\ \text{mg}}{n_{\text{rotor}} n_{\text{blade}} \bar{c} \bar{c}_l \rho r}}$$

Likewise, the disk solidity ratio σ is defined as

$$\sigma = \frac{n_{\text{blade}} \bar{c}}{\pi r}$$

The blade drag coefficient c_{bd} depends primarily on the airfoil and typically increases with the blade lift coefficient and thrust coefficient. For simplicity, this parameter is set constant as $c_{bd} = 0.075$.

Finally, the downwash w is determined by solving

$$\frac{T}{2\rho R} = w\sqrt{(w - v_a \sin\alpha)^2 + (v_a \cos\alpha)^2}$$

whereby α is the angle of attack, which is calculated by

$$\alpha = \arctan\left(\frac{-D_{\text{body}} - mg\sin\gamma}{mg\cos\gamma}\right)$$

Acknowledgements

This work is partially funded by the Engineering and Physical Sciences Research Council project CASCADE under grant number EP/R009953/1. The authors would like to extend thanks for the support from the project and the great collaboration with the project partners.

References

[1] Toth, P., and Daniele, V., "The Vehicle Routing Problem," *An Overview of Vehicle Routing Problem*, Vol. 9, Soc. for Industrial and Applied Mathematics, Philadelphia, PA, 2002, pp. 1–26.

[2] Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S., "Vehicle Routing Problems for Drone Delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 47, No. 1, 2017, pp. 70–85. https://doi.org/10.1109/TSMC.2016.2582745

[3] Hong, I., Kuby, M., and Murray, A. T., "A Range-Restricted Recharging Station Coverage Model for Drone Delivery Service Planning," *Transportation Research Part C: Emerging Technologies*, Vol. 90, Feb. 2018, pp. 198–212. https://doi.org/10.1016/j.trc.2018.02.017

[4] Ebongue, J., Bayaola, I., Thron, C., and Förster, A., "Charging Stations Placement in Drone Path Planning for Large Space Surveillance," *CARI 2020 - Colloque Africain sur la Recherche en Informatique et en Mathématiques Apliquées*, Thies, Sénégal, Oct. 2020.

[5] Lin, J., Zhou, W., and Wolfson, O., "Electric Vehicle Routing Problem," *Transportation Research Procedia*, Vol. 12, Jan. 2016, pp. 508–521. https://doi.org/10.1016/j.trpro.2016.02.007

[6] Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., and Guizani, M., "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, Vol. 7, April 2019, pp. 48572–48634. https://doi.org/10.1109/ACCESS.2019.2909530

[7] Erdelic, T., Carić, T., and Lalla-Ruiz, E., "A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches," *Journal of Advanced Transportation*, Vol. 2019, 2019, pp. 1–27. https://doi.org/10.1155/2019/5075671

[8] Kool, W., van Hoof, H., and Welling, M., "Attention, Learn to Solve Routing Problems!" Preprint, submitted 22 March 2018, https://arxiv.org/abs/1803.08475.

[9] Yu, J. J. Q., Yu, W., and Gu, J., "Online Vehicle Routing with Neural Combinatorial Optimization and Deep Reinforcement Learning," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20, No. 10, 2019, pp. 3806–3817. https://doi.org/10.1109/TITS.2019.2909109

[10] Mausam, A., and Kolobov, A., *Planning with Markov Decision Processes: An AI Perspective, Synthesis Lectures on Artificial Intelligence and Machine Learning*, Springer, New York, 2012, pp. 7–27. https://doi.org/10.2200/S00426ED1V01Y201206AIM017

[11] Tahami, H., Rabadi, G., and Haouari, M., "Exact Approaches for Routing Capacitated Electric Vehicles," *Transportation Research, Part E: Logistics and Transportation Review*, Vol. 144, Nov. 2020, Paper 102126. https://doi.org/10.1016/j.tre.2020.102126

[12] Desaulniers, G., Errico, F., Irnich, S., and Schneider, M., "Exact Algorithms for Electric Vehicle-Routing Problems with Time Windows," *Operations Research*, Vol. 64, No. 6, 2016, pp. 1388–1405. https://doi.org/10.1287/opre.2016.1535

[13] Kucukoglu, I., Dewil, R., and Cattrysse, D., "The Electric Vehicle Routing Problem and its Variations: A Literature Review," *Computers and Industrial Engineering*, Vol. 161, July 2021, Paper 107650. https://doi.org/10.1016/j.cie.2021.107650

[14] Ruland, K. S., and Rodin, E. Y., "The Pickup and Delivery Problem: Faces and Branch-and-Cut Algorithm," *Computers and Mathematics with Applications*, Vol. 33, No. 12, 1997, pp. 1–13. https://doi.org/10.1016/S0898-1221(97)00090-4

[15] Ropke, S., and Cordeau, J. F., "Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows," *Transportation Science*, Vol. 43, No. 3, 2009, pp. 267–286. https://doi.org/10.1287/trsc.1090.0272

[16] Chiang, W. C., and Russell, R. A., "Simulated Annealing Metaheuristics for the Vehicle Routing Problem with Time Windows," *Annals of Operations Research*, Vol. 63, Feb. 1996, pp. 3–27. https://doi.org/10.1007/BF02601637

[17] Homberger, J., and Gehring, H., "Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows," *INFOR: Information Systems and Operational Research*, Vol. 37, No. 3, 1999, pp. 297–318. https://doi.org/10.1080/03155986.1999.11732386

[18] de Oliveira da Costa, P. R., Mauceri, S., Carroll, P., and Pallonetto, F., "A Genetic Algorithm for a Green Vehicle Routing Problem," *Electronic Notes in Discrete Mathematics*, Vol. 64, Feb. 2018, pp. 65–74. https://doi.org/10.1016/j.endm.2018.01.008

[19] Rastani, S., and Çatay, B., "A Large Neighborhood Search-Based Matheuristic for the Load-Dependent Electric Vehicle Routing Problem with Time Windows," *Annals of Operations Research*, Springer, New York, 2021, pp. 1–33. https://doi.org/10.1007/s10479-021-04320-9

[20] Ghilas, V., Demir, E., and Van Woensel, T., "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines," *Computers and Operations Research*, Vol. 72, Aug. 2016, pp. 12–30. https://doi.org/10.1016/j.cor.2016.01.018

[21] Goeke, D., "Granular Tabu Search for the Pickup and Delivery Problem with Time Windows and Electric Vehicles," *European Journal of Operational Research*, Vol. 278, No. 3, 2019, pp. 821–836. https://doi.org/10.1016/j.ejor.2019.05.010

[22] Vinyals, O., Fortunato, M., and Jaitly, N., "Pointer Networks," 2015, pp. 1–9. https://doi.org/10.48550/ARXIV.1506.03134

[23] Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S., "Neural Combinatorial Optimization with Reinforcement Learning," *5th*

International Conference on Learning Representations, ICLR 2017—Workshop Track Proceedings, 2016, pp. 1–15.

[24] Nazari, M., Oroojlooy, A., Takáč, M., and Snyder, L. V., "Reinforcement Learning for Solving the Vehicle Routing Problem," *Advances in Neural Information Processing Systems*, Vol. 2018, Dec. 2018, pp. 9839–9849.

[25] Bono, G., Dibangoye, J. S., Simonin, O., Matignon, L., and Pereyron, F., "Solving Multi-Agent Routing Problems Using Deep Attention Mechanisms," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 27, No. 1, 2020, pp. 1–10. https://doi.org/10.1109/tits.2020.3009289

[26] Lin, B., Ghaddar, B., and Nathwani, J., "Deep Reinforcement Learning for Electric Vehicle Routing Problem with Time Windows," Preprint, submitted 5 Oct. 2020, https://arxiv.org/abs/2010.02068.

[27] Li, J., Xin, L., Cao, Z., Lim, A., Song, W., and Zhang, J., "Heterogeneous Attentions for Solving Pickup and Delivery Problem via Deep Reinforcement Learning," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 3, 2021, pp. 1–10. https://doi.org/10.1109/TITS.2021.3056120

[28] Gong, L., and Cheng, Q., "Exploiting Edge Features for Graph Neural Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Inst. of Electrical and Electronics Engineers, New York, 2019, pp. 9203–9211. https://doi.org/10.1109/CVPR.2019.00943

[29] Wang, Z., Chen, J., and Chen, H., "EGAT: Edge-Featured Graph Attention Network," *Artificial Neural Networks and Machine Learning–ICANN 2021*, *Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence, and Lecture Notes in Bioinformatics*, Vol. 12891, Springer, New York, 2021, pp. 253–264. https://doi.org/10.1007/978-3-030-86362-3_21

[30] Bono, G., "Deep Multi-Agent Reinforcement Learning for Dynamic and Stochastic Vehicle Routing Problems To Cite this Version : HAL Id : tel-03098433 Deep Multi-Agent Reinforcement Learning for Dynamic and Stochastic Vehicle Routing Problems," Universit'e de Lyon, 2020.

[31] Basso, R., Kulcsár, B., and Sanchez-Diaz, I., "Electric Vehicle Routing Problem with Machine Learning for Energy Prediction," *Transportation Research, Part B: Methodological*, Vol. 145, March 2021, pp. 24–55. https://doi.org/10.1016/j.trb.2020.12.007

[32] Wang, B. H., Wang, D. B., Ali, Z. A., Ting Ting, B., and Wang, H., "An Overview of Various Kinds of Wind Effects on Unmanned Aerial Vehicle," *Measurement and Control (United Kingdom)*, Vol. 52, Nos. 7–8, 2019, pp. 731–739. https://doi.org/10.1177/0020294019847688

[33] Zhang, J., Campbell, J. F., Sweeney, D. C., and Hupman, A. C., "Energy Consumption Models for Delivery Drones: A Comparison and Assessment," *Transportation Research, Part D: Transport and Environment*, Vol. 90, Dec. 2020, Paper 102668. https://doi.org/10.1016/j.trd.2020.102668

[34] Kirschstein, T., "Comparison of Energy Demands of Drone-Based and Ground-Based Parcel Delivery Services," *Transportation Research Part D: Transport and Environment*, Vol. 78, Dec. 2019, Paper 102209. https://doi.org/10.1016/j.trd.2019.102209

[35] Langelaan, J. W., Schmitz, S., Palacios, J., and Lorenz, R. D., "Energetics of Rotary-Wing Exploration of Titan," *IEEE Aerospace Conference Proceedings*, IEEE Publ., Piscataway, NJ, 2017, pp. 1–11. https://doi.org/10.1109/AERO.2017.7943650

[36] Kimon, P., and Valavanis, G. J. V., *Handbook of Unmanned Aerial Vehicles*, Vol. 1, Springer, 2015, pp. 307–329. https://doi.org/10.1007/978-90-481-9707-1_47

[37] Johnson, G. L., *Wind Energy Systems*, Prentice–Hall. Englewood Cliffs, NJ, 1985. https://doi.org/10.1109/JPROC.2017.2695485

[38] Moorhouse, D. J., and Woodcock, R. J., "Background Information and User Guide for MIL-F-8785B, Military Specification—Flying Qualities of Piloted Airplanes," *Mil-F-8785C*, Air Force Wright Aeronautical Labs, Wright-Patterson AFB OH, 1982, p. 255, http://www.dept.aoe.vt.edu/~durham/AOE5214/MILSPEC8785C.pdf.

[39] Hussain, M. S., Zaki, M. J., and Subramanian, D., "Global Self-Attention as a Replacement for Graph Convolution," *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, Aug. 2022. https://doi.org/10.1145/3534678.3539296

[40] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., *Attention Is All You Need*, 2017. https://doi.org/10.48550/ARXIV.1706.03762

M. J. Kochenderfer
Associate Editor

Edge-enhanced attentions for drone delivery in presence of winds and recharging stations

Liu, Ruifan

2023-04-01

Liu R, Shin H-S, Tsourdos A. (2023) Edge-enhanced attentions for drone delivery in presence of winds and recharging stations. Journal of Aerospace Information Systems, Volume 20, Issue 4, April 2023, pp. 216-228

https://doi.org/10.2514/1.I011171