In: AIAA Aviation 2022 Forum, 27 June - 1 July 2022, Chicago, IL. Paper number AIAA 2022-4028
DOI: 10.2514/6.2022-4028

Delivery Route Planning for Unmanned Aerial System in Presence of Recharging Stations

Ruifan Liu[1], Hyo-sang Shin[2], Miuguk Seo[3], Antonios Tsourdos[4]
Cranfield University, Bedfordshire, MK43 0AL, United Kingdom

Existing variants of vehicle routing problems (VRPs) have limited capability 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 model 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 (MDP) 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. AM-E comprises a succession of edge-enhanced dot-product attention layers which is designed with the aim of capturing the heterogeneous node relationship for DDP-Rs by incorporating adjacent edge information. Simulation shows 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 non-wind and windy cases, extensive simulations demonstrate that the proposed DRL method outperforms state-of-the-art heuristics for solving DDP-R, especially at large sizes.

I. Introduction

Rapid developments in drone technology 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 began to authorize flights beyond visual line of sight (BVLOS) to test the safety of opening up the technology to the wider industry. Integrating drones into ITS 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 modelling multi-vehicle logistics with the objective of seeking the best route for a fleet of vehicles to serve a set of customers. The short duration of a Li-Po battery-equipped drone courier could be met by modelling the delivery as *Capacitated Vehicle Routing Problem* (CVRP), a variant of VRP that describes scenarios where a vehicle starts and ends multiple routes at one common depot without exceeding its limited carrying capacity. However, in CVRP, the drone must continuously return to a common depot to renew its battery, resulting in low efficiency. Deploying recharging facilities expands the mission execution area and is also eco-friendly. The routing problem considering recharging is termed as the *Electric Vehicle Routing Problem* (EVRP) [1], also known as the *Green Vehicle Routing Problem*, and was 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 centre, a delivery request is likely to have its own pickup and delivery locations,

[1] PhD candidate, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University.
[2] Professor, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University.
[3] Research fellow, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University.
[4] Professor, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University.

such as those in a medical delivery service [2]. This type of route planning problem can be described as *Pickup and Delivery Problems* (PDP), another variant of VRP characterized by pairing and precedence relationships. To address the unmanned aerial logistics while both considering recharging facilities and heterogeneous pickup and delivery requirements, a combination of EVRP and PDP is investigated in this paper. Moreover, compared to ground-based 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-hard and stochastic nature, conventional methods including exact and heuristic algorithms struggle to efficiently address this routing problem[3]. Recently, there has been increasing attention on *Deep Reinforcement Learnin*g (DRL) for solving combinatorial optimization problems, which has delivered promising results on the basic VRP and some of its variants. To this end, this paper explores learning techniques for solving the drone delivery problem considered.

Since 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 above-mentioned drone delivery problem. Firstly, 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. Secondly, 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 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 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 RL method. Through the comparison of state-of-the-art algorithms, the proposed DRL-based planning method is shown to provide high-quality solutions (within a 2% gap to optimal solutions) and demonstrates robustness in the presence of stochastic winds.

2) A policy neural network called *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 elaborate node connectivity extensively without increasing its computational complexity.

3) Compared to existing works, the drone delivery problem studied in this paper is formulated from a more practical perspective considering pickup-delivery tasks, the effect of varying wind, and the presence of recharging stations. After describing the problem in the form of *Markov Decision Processes* (MDPs), a good policy is sought by devising the evolution of the environment, masking scheme, and reward function.

The rest of the paper is organized as follows: Section 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 drones' energy consumption model. This is followed by the proposed DRL-based in Section 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. Section VI 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

Since the VRP is an NP-hard problem, either EVRP or PDP is a generalization of the VRP, thus are equally considered NP-hard in the strong sense [4]. Due to the complexity of the problem, studies using exact approaches are rarely found in literature, and most of them adopt branch-and-bound algorithms or their variants. A branch and cut algorithm is proposed in [5] to solve the EVRP. Similarly, a branch-price-and-cut algorithm was used in [6] to solve the EVRP with time windows. Various branch-and-price algorithms were designed, targeting specific extended versions of EVRP, e.g., with heterogeneous stations, partial charging policy, and non-linear charging rate [7]. In terms of PDP, extra bounding procedures or column generation schemes are introduced to the basic branch-and-bound

algorithm to meet the pickup-delivery constraints [8][9]. 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 to tackle the problem approximately but efficiently. Typical metaheuristics include neighbourhood search, simulated annealing [10], tabu search, evolutionary metaheuristics [11] and genetic algorithms [12]. Rastani *et al.* [13] integrated an optimal repair procedure in a large neighbourhood search heuristic method for solving the load-dependent EVRP with time windows. Another adaptive large neighbourhood search method was proposed in [14] to solve PDP based on the destroy and re-create principle. Particularly, stated as the first solution to PDP with electric vehicles, a granular tabu search algorithm is developed in [15] by restricting the neighbourhood of 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, since they fail to efficiently deal with the unforeseen changes, i.e., incoming requests. In terms of stochastic routing problems, the problem is modelled 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 to employ 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 Seq2Seq networks are found in the literature to map the state space to action probabilities.

The Pointer Network (PtrNet) is the first seminar work to cope with the routing problem via learning, which solves TSP via recurrent neural networks (RNN) [16]. The PtrNet is firstly implemented in a supervised manner and later extended into an RL framework [17]. Furthermore, the work in [18] generalizes the application of PtrNet to a wider range of CO such as CVRP, where element embeddings could be dynamic. After that, more effective deep learning (DL) architectures are proposed to represent problem statements by combining the transformer-based attention model [19][20] or graph embedded structure [21][22], showing outperforming results in comparison with heuristic methods. A stage has arrived where deep neural networks (DNNs) 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 pickup and delivery problem (PDP) or electric vehicle routing problem (EVRP), the above-mentioned networks could be directly employed with 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.* [23] propose a heterogeneous attention model for PDP, in which seven types of attention layers are sophisticatedly designed to consider different roles played by nodes while taking the precedence constraint into account. Lin *et al.* [22] incorporate the model of [18] with a graph-embedding component to yield the global information of the graph for EVRP with time windows. Nevertheless, most solutions of PDP and EPDP are still focusing on heuristic approaches, which heavily depend on human expertise, leaving much room for improving solution quality.

On the other hand, the routing problem with stochastic model parameters has received increasing attention. Instead of assuming 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. Since RL provides promising tools to optimize policies with stochastic state transitions, Bono *et al.* [24] 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 [25], achieving more energy-saving and reliability for routes.

III.Problem Statement

This section formulates the *drone delivery problem with recharging* (DDP-R) and introduces a stochastic drone energy consumption model in consideration of varying wind conditions.

A. Drone delivery problem with recharging

Suppose a drone delivery scenario where there are □ customer requests and each of them is decomposed into a pickup task □□ □ and a delivery task □□ □ □ □. 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 □□ □. The aim of solving the drone delivery problem is to find a route that minimizes the energy

cost while subjecting to constraints regarding the delivery mission and drone properties. Further assumptions used in this study are listed below:

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

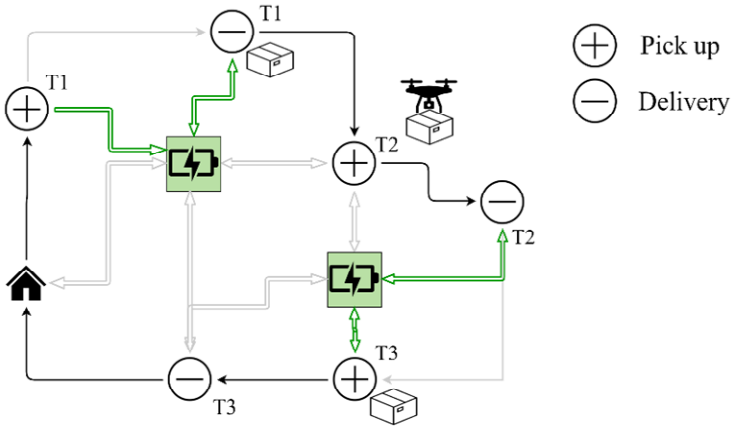

Fig. 1. Drone delivery with recharging

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 as they directly impact the flight kinematic model. Nonetheless, most route optimization models only incorporate energy consumption implicitly via a pre-defined 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, inspired by the work in [26] and [27].

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 [27]:

$$P_{aero} = P_{parasite} + P_{induced} + P_{profile} + P_{climb}$$

$$P_{aero} = \frac{1}{2}\rho V^3 C_{D_0} A + (1+k)\frac{W^2}{\rho V A} + \frac{\sigma C_{d_0}}{8}\rho A \Omega^3 R^3 + W V \sin\gamma$$

where V is the airspeed, γ is the flight angle, k is induced factor (usually equal to 1.15, see [27]), body drag force $D = \frac{1}{2}\rho V^2 C_{D_0} A$, thrust $T = \sqrt{\left(\frac{W^2}{\rho V A}\right)^2 + \left(\frac{1}{2}\rho V^2 C_{D_0} A\right)^2}$, κ is the downwash coefficient, A is the rotor dis area, ΩR is the blade tip speed, σ is the rotor solidity ratio and C_{d_0} is the blade drag coefficient. Details about determining κ, A, ΩR, σ and C_{d_0} are given in Appendix B along with an overview of notations.

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

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

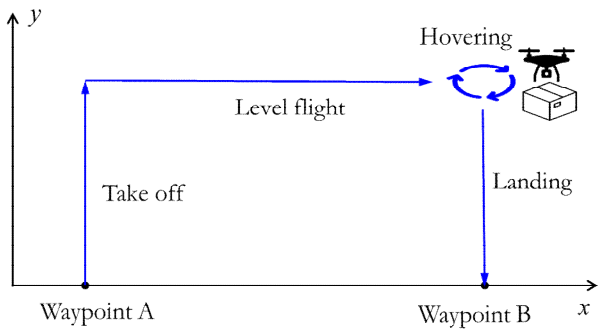

Fig. 2. The flight profile for the VTOL drone delivery, consisting of four phases: takeoff, level flight, hovering and landing.

For delivery problems, we identify the flight process as having four phases: take-off, 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:

$$\square \square \frac{1}{\tau}\square\square_{\square\square\square}\square_{\square\square\square\square}\square\square \square\square^{\square\square\square}\square^{\square\square\square}\square\square \square_{\square\square}\square_{\square\square\square\square}\square\square \square\square^{\square\square}, 0^{\square}\square\square \square_{\square\square\square\square}\square_{\square\square\square\square}\square\square \square]\square_{\square} |, 0^{\square})$$

$$\square \square_{\square\square\square}\square_{\square\square\square\square}\square\square \square\square^{\square\square\square\square}\square^{\square\square\square\square}\square\square \square \square_{\square\square\square\square}\square_{\square\square\square\square}$$

Where $]\square_{\square} |$ is the wind speed, $\square_{\square\square\square}$, $\square_{\square\square\square\square}$ indicates the time for takeoff and landing, $\square_{\square\square\square\square\square}$ is time for hovering, $\square_{\square\square}$ is time for level flight, and $\square_{\square\square\square\square\square}$ is the total duration. In this paper, we consder the drone with vertical take-off and landing (VTOL), indicating that the ascent angle $^{\square}\square\square\square = 90^{\square}$ and the descent angle $\square_{\square\square\square\square} \square \square\square\square^{\square}$.

Given the altitude of level flight $\square$ and the UAV ground speed $\square_{\square}$ ($\square_{\square} \square \square_{\square}$ with no wind), the time for level flight is $\square_{\square\square} = \frac{\square}{\square_{\square}}$, and the time for takeoff and landing is $\square_{\square\square\square} \square \square_{\square\square\square\square} = \frac{\square}{\square_{\square}}$, where $\square_{\square}$ is the vertical speed.

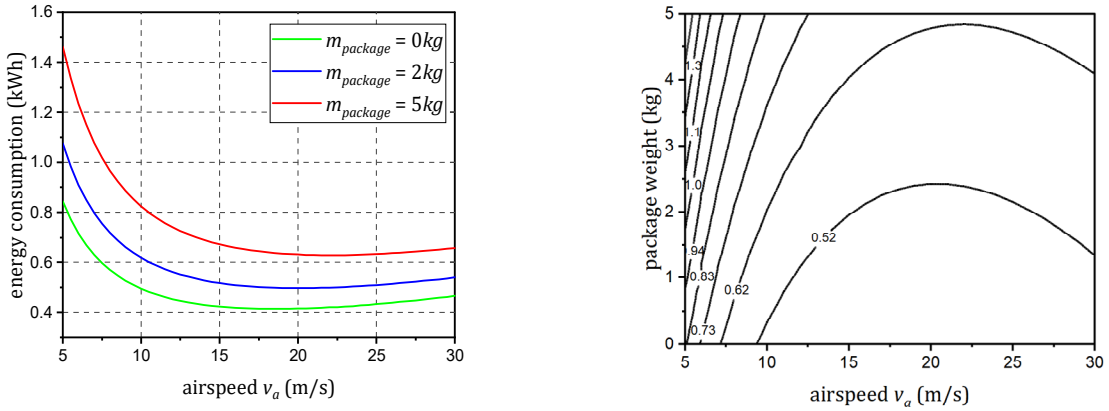

Fig. 3. Energy consumptions for a 10km flight with airspeed and package weight varying. In the first figure, three cases are considered with package weight $\square_{\square\square\square\square}$ $\square$ $\square\square\square\square\square\square\square\square\square\square$ respectively. In the second figure, a contour map of energy consumption is depicted with regard to airspeed and package weight. In simulations, no wind is considered, altitude for level flight $\square\square$ $\square\square\square\square$, hovering time $\square_{\square\square\square\square\square}$ $\square$ $\square\square$, and vertical speed is assumed as a constant $\square_{\square}$ $\square$ $\square\square$ c$\square$

2. Energy consumption in presence of wind

According to the above model, energy cost for a specific delivery is defined by three variables: package weight $\square_{\square\square\square\square}$, airspeed $\square_{\square}$, and the flight time $\square$ Further assume that the drone is directed by a constant ground speed

command, the airspeed is then obtained by integrating the velocity of wind and the ground speed command, which is derived from the drone kinematic model:

$$\dot{x} = v_a \cos\psi + v_w \cos\psi_w$$
$$\dot{y} = v_a \sin\psi + v_w \sin\psi_w$$

where v_g, v_a and v_w indicate the ground speed, airspeed and wind speed respectively, ψ UBC$^{\varphi}$ c 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

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

In terms of wind modelling, we establish a stochastic wind model in which a constant wind is combined with a random turbulence flow. The constant wind speed is randomly generated via a Weibull distribution, which is widely used to describe wind variation [28]. The scale parameter and shape parameter in wind Weibull distributions indicate the mean value of wind speed and distribution shape, respectively. The direction of the constant wind is assumed to follow a uniform distribution from $\psi_w = 0^{\circ}$ to $\psi_w = 360^{\circ}$. The turbulence wind component is generated through the well-known Von Karman turbulence model [29].

To analyze the distribution of energy consumption under varying wind conditions, we conduct a series of Monte Carlo simulations given a single-way waypoint task, where the distance to the target position is 10 , and the flight is unloaded. After running simulations of 10000 rounds each with different airspeeds and average wind speed, the statistical figure of the energy consumption is depicted in Fig 4.

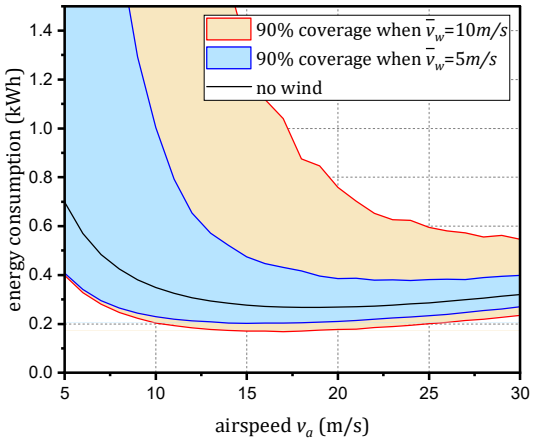

Fig. 4. Statistical distribution of energy consumption for a flight with the wind. In Monte Carlo simulations, the shape parameter of the wind Weibull distribution is set as .

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 c or c. As shown from the figure, these two shaded areas are both unignorable. We can also find out that when airspeed increases, the UAV could resist the wind to some degree, leading to a smaller variance. However, even with c airspeed, winds having an average speed c could also cause a difference in the energy cost up to for a path segment, and the deviation even becomes 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. This emphasizes the importance to introduce the stochastic energy consumption model into planning to make the results more wind resistant.

IV. Reinforcement learning model

This section describes details about the formulation of the stochastic drone delivery problem with recharging (stochastic DDP-R) in the form of MDPs and the design of a DRL-based algorithm to solve the problem formulated.

The architecture of the policy network, the masking policy and the RL training method are also introduced in this section.

A. 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 □, action space □, immediate reward function □□□□□□ and state transition probability □□□□|□□□□ Specifically, for the DDP-R described above, these components are defined as follows:

State: The global state □□ □ □ is factored into the drone state □□ (location, remaining energy) and the mission state □□ including locations and status of customer requests, recharging stations and the depot. The mission state is further decomposed into the vertex features □□ (location) and the edge features □□ (distance, connectivity).

□ □ □□□ □□□ □ □□□□□□□□ □ □

where

□□ □ ⟨□□ □□□ □□□□⟩
□□ □ □□□ □□□ □□□
□□ □ □□□ □□□

where □□□ □□□) and □□□ □□□ is the coordinate of the drone and the vertex □, □□□ is the state of charge level of the drone, □□ is the weight of the package, □□□ is the distance between vertex □ and □, and □□□ is a binary variable, indicating directional connectivity from vertex □ to vertex □

It is noted that there are two possible route ends:

a. All customer requests have been served.
b. 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 □□ □ □□.

Action: The action indicates the next movement of the UAV, represented by the corresponding vertex: □ □ □□□□ □. 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 Section 5.3.

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 problem 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 □ for the drone traversing an edge □, which depends on 1) the payload □, 3) wind conditions □. Thus, the reward associated with each travel is set as the negative energy cost of the travel under the configuration □□□□ □

□□□□□□□ □ □□□□□□□ □

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 □□□□ is imposed for every task that is left unfinished at the end of the episode, which is denoted as a terminal reward □□:

$$r_T = \begin{cases} -p_{\text{unserved}} \cdot n_{\text{unserved}} & \text{if } v_t = v_0 \\ 0 & \text{otherwise} \end{cases}$$

where □□□□ counts the number of pending tasks when the state arrives at □□.

Transition: A transition function describes how states are updated, represented by the probability □□□]□□□□that the current state □is converted to the next state □by taking the action □. When the UAV travels to the vertex □after choosing action □ □ □, the drone's state and the mission state get updated according to the type of node the drone visited.

1) 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, using the model described in Section 3.2;

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

B. Edge-enhanced attention model

The policy network adapts the original *Attention Model* (AM) [19] for a better description of the considered DDP-Rs using a *dot-product attention layer with edges*, which is inspired by embedding techniques in *Graph Neural Network* (GNN). Most policy networks, as discussed in Section 2, were designed for homo-vertex 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, which together with the distance matrix forms edge features. The adjacent matrix is constituted of binary element □□ which describes the connectivity from the node □to the node □ For further elaboration, a toy example is given in Fig. 5, which contains 2 delivery tasks, 1 depot and 1 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, while the first column indicates the vehicle can only return back to the depot from the delivery destinations and rechargings. In terms of recharging stations, they are indirectly connected with all other nodes except from themselves. For pickup nodes, they are only connected to their corresponding delivery locations and recharging facilities, while for delivery tasks, they could direct to any other pickup positions with the the empty vehicle except from its corresponding pickup task since the parcel has just been delivered.

To summarize, the logic behind the connectivity matrix comprises: the edge between paired requests is a one-way flight from pickup locations to delivery locations, the drone is not able to load another parcel while loaded, and the empty drone is not expected to fly to any delivery destinations.

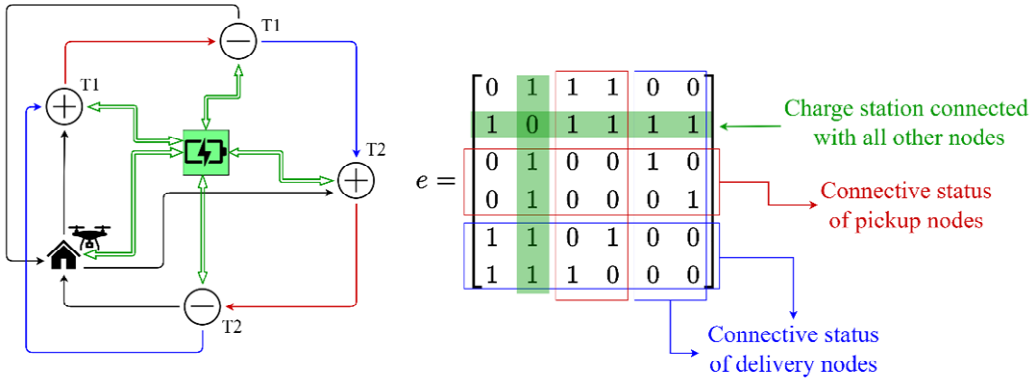

Fig. 5. An example of the adjacent matrix in DDP-Rs. According to delivery logic and payload constraint, the connectivity matrix is defined as shown in the figure.

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* (Fig.7) and integrate it into the Attention Model.

1. Edge-enhanced dot-product attention layer

Exploring edge information in the graph neural network, this paper is not the first trial. Gong *et al.* [30] have built a generic framework to sufficiently exploit edge features. The framework can consolidate both *graph convolutional networks* (GCN) and *graph attention networks* (GAT), extending the one-dimensional adjacent matrix to multi-dimensional edge features. Based on the idea of [30], Wang *et al.* [31] and Hussain *et al.* [32] instantiates the edge augmentation with GAT and the transformer network respectively.

Though the intention of using extra edge features is to enhance the performance, these edge-integration networks do not always perform superiorly. Simulations in [31] revealed that *edge-featured* GAT (EGAT) even has a slight performance degradation compared to GAT when executing node-sensitive tasks (e.g., *Cora, Citeseer*) though achieving higher accuracy on edge-sensitive tasks (e.g., trading network classifications). As analyzed in [31], this is due to the fact that in 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. The EGAT model in [31] couplingly embeds the node features □ and the edge feature □ by each GAT layer, while the proposed EGAT only updates the node features retaining edge features 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.

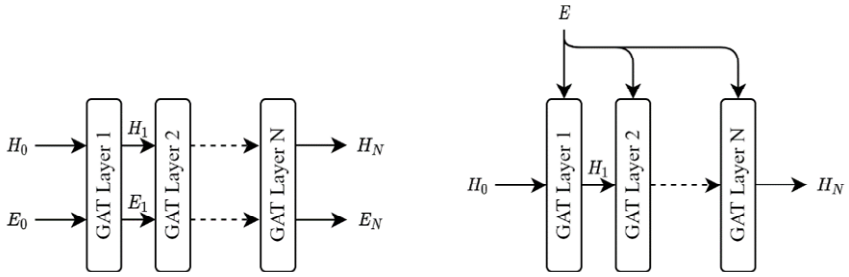

Fig. 6. Two edge-enhanced GAT architectures. (add more explanation)

The attention mechanism performs a message passing process over nodes of a graph, during which weights are added when integrating neighbourhood 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 neighbours □, and weights them using the *compatibility* of its *query* □ with the *key* of the neighbour □. To take advantage of edge information, we integrate the embedded edge feature into the *compatibility* and also merge it into the *value* of neighbours, as illustrated in Fig.7.

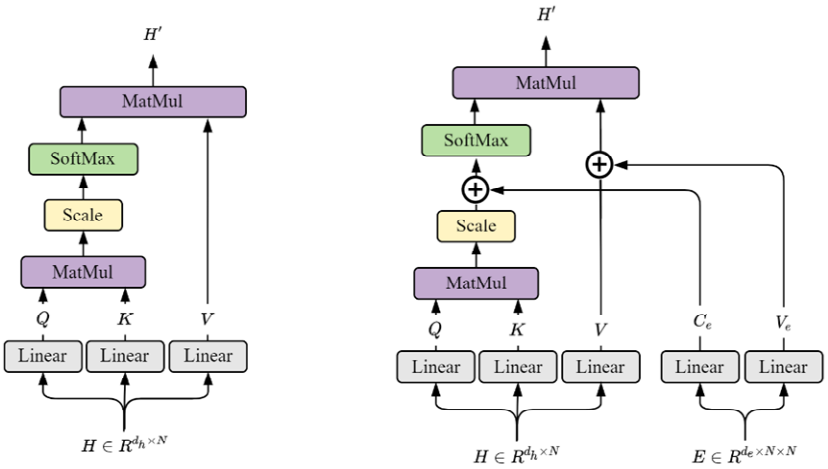

Fig. 7. (left) Scaled dot-product attention. (right) Edge-enhanced scaled dot-product attention. In edge-enhanced scaled dot-product attention, edge features are projected and then merged into the original *value* vector and the *compatibility* vector, i.e, the scaled dot-product of *query* and *key*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

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

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

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} are also linearly projected, constructing the edge component for the *compatibility* and the *value* of each edge:

$$e^{c}_{ij} = W^{C} e_{ij} \quad e^{v}_{ij} = W^{V} e_{ij}.$$

Here learnable parameter $W^{C} \in \mathbb{R}^{1 \times d_e}$ is used to generate a 1-dimensional *edge compatibility* matrix e^{c}_{ij}, and $W^{V} \in \mathbb{R}^{d_v \times d_e}$ 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:

$$a_{ij} = \mathrm{softmax}\left(\frac{q_i k_j}{\sqrt{d_k}} + e^{c}_{ij}\right)$$

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

$$h'_i = \sum_{j} a_{ij}\left(v_j + e^{v}_{ij}\right)$$

To enhance the capability of expression and also benefit the stability of the attention learning process, the single-layer attention is then extended to the *multi-head 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 multi-head attention value for the node is:

$$h'_i = h_i + \mathrm{BN}\left(\sum_{m=1}^{M} W^{O,m} h'^{(m)}_i\right)$$

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

2. The encoder-decoder architecture

Fig. 8 depicts the overall architecture of our policy network, called *Attention Model with Edges* (AM-E). AM-E inherits the encoder-decoder structure of the Transformer model, which is recoginized as the most competitive neural sequence transduction model [33]. Taking a toy instance (2 deliver requests, 1 recharging station, and 1 depot) for example, the vertex features $[h^{0}_{0} \quad \cdots \quad h^{0}_{3}]$ and edge features e^{0} are first passed to the dash line-blocked attention module after being initially embedded. 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 $[h^{N}_{0} \quad \cdots \quad h^{N}_{3}]$. Afterwards, using drone features h^{p} as the *query* vector and masking unavailable nodes, we could finally get the policy output $[p_0 \quad \cdots \quad p_3]$, which is the policy probability of each node.

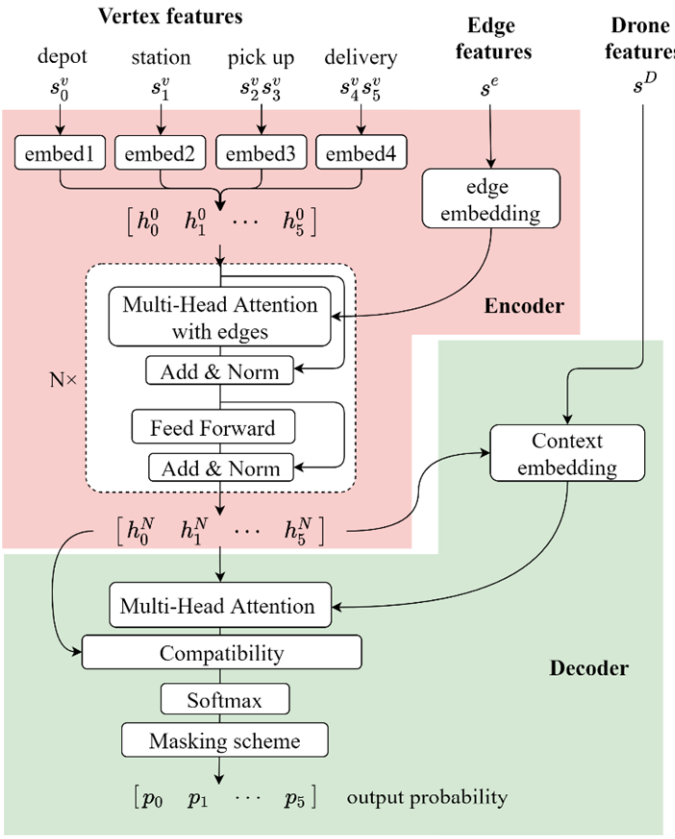

Fig. 8. AM-E policy network

Regarding the encoder, a similar structure to the Transformer [33] is adopted, which provides a mapping from nodes' raw features to a richer embedding space, through which the node's own features, the features of its neighbours, and the features of edge connected with neighbours are all represented. Specifically, mission features $[\square\square\square]$ are initially embedded to a larger dimension $\square_\square$ via the node-wise 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 $\square_\square^{(\square)}$ are recursively updated by $\square$ multi-head attention layers, each consisting of three operations: multi-head attention, feed forward and normalization (see [19] 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 $\square$ for selecting nodes at each step. To achieve this, the graph is augmented with a special *context node* $\square\square\square$ 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:

$$\square_{(\square)}^{(\square)} \; \square \; \square\square_{(\square)}^{(\square)}\square\square_{(\square)}\square$$

where the context $\square_{(\square)}$ is the concatenation of an aggregated node embedding $\square_{(\square)}^{(\square)}$:

$$\square_{(\square)}^{(\square)} = \frac{1}{\square}\sum_{\square\square\varphi}^{\square} \square_\square^{(\square)}$$

and $\square_{(\square)}$ is the linear projection of the drone state $\square^\square$:

$$\square_{(\square)} \; \square \; \square^{(\square)}\square^\square \; \square \; \square^{(\square)}.$$

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

$$h_{(c)}^{(N+1)} = \mathrm{MultiHeadAttention}\left(W^Q h_{(c)}, W^K h^{(N)}, W^V h^{(N)}\right)$$

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 $q_{(c)} = W^Q h_{(c)}^{(N+1)}$ and $k_i = W^K h_i^{(N)}$. The compatibility is then clipped within $[-C, C]$ using a $\tanh$, which is followed by the masking policy described in Section 4.3:

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

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

$$p_i = p_\theta(\pi_t = i \mid s, \pi_{1:t-1}) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}}$$

C. Masking scheme

With the purpose to improve exploration efficiency and ensure 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 $a \in A_t$ is labelled as valid only if all of the following conditions are satisfied:

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

D. Training method

Shown as Algorithm1, we train the policy network using the REINFORCE algorithm with baselines. The baseline is chosen as rollout or critic according to whether the problem is deterministic or stochastic. In detail, at the beginning of each episode, one batch of new samples is stochastically generated. Then routes are sequentially sampled according to the probability output from the policy network and rewards are collected. Moreover, we get the expected reward $b(s)$ 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 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.

Algorithm 1 REINFORCE learning algorithm
Input: number of epochs E, batch size B
foreach training epoch $e = 1,2,\dots,E$ **do**
generate instances of batch size B;

foreach instance $= 1,2, \ldots, B$ **do /* ran in parallel */**
  sample trajectory using policy p_θ;
  receive the accumulative reward of the trajectory L_i;
  if *rollout baseline* **then**
    receive baseline reward L^{BL}_i using GreedyRollout policy $p_{\theta^{BL}}$;
  else if *critic baseline* **then**
    estimate values of initial states using critic b_v;
  end
end
estimate mean Policy Gradient on the batch of trajectories:
$\nabla_\theta J(\theta) \approx \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log p_\theta(\pi_i) [L_i - L^{BL}_i]$);
if *rollout baseline* **then**
  update policy parameters θ;
  if $\sum_{i=1}^{N} L_i - \sum_{i=1}^{N} L^{BL}_i < \alpha$ **then**
    replace $p_{\theta^{BL}}$ using θ;
  end
else if *critic baseline* **then**
  estimate the mean gradient of the MSE of critic on the batch of trajectories:
  $\nabla_{\theta_v} = \frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta_v} \| b_v - L_i \|^2$;
  update policy parameters θ and critic parameters θ_v;
end
end

V. Numerical Simulations

To verify the effectiveness and evaluate the performance of the proposed AM-E model, we conduct simulations for solving DDP-R problems 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, pickup tasks, delivery tasks, charging stations, and the depot are randomly located on a $\square\square\square\square \times \square\square\square\square$ 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 $\square\square\square$ c and a battery capacity of $\square\square\square\square\square$ is assumed. The energy cost on each edge is decided by the energy consumption model presented in Section 3.2, 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 Section 3.2 and assumed to remain unchanged for each path segment.

2) Network Structure: For the proposed AM-E model, the node embedding, edge embedding, and context embedding are all one-layer element-wise liner projections with dimension 128. The multi-head graph attention network consists of $\square$ $\square$ $\square\square$heads computing *key* vectors and *value* vectors of dimension $d_k = d_v = 16$. The number of sequential multi-attention modules is 3. In the feed-forward layer, the node features are passed through node-wise projections of one hidden sublayer with the ReLu activation and 512 hidden units.

3) Training Parameters: We adopt the Adam Optimizer to train the network with a constant learning rate $\alpha = 10^{-4}$. 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 mins for DDP20-R (K80), 42 mins for DDP40-R (K80), and 28 mins for DDP80-R (V100).

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 model, the original Attention Model (AM) [19], and the heterogeneous

attention model [23]. As mentioned, the heterogeneous attention model is designed for *pickup and delivery problems* (PDP) featuring a better performance by considering heterogeneous roles in PDP. It does not exactly match the DDP-R studied in this paper because of the presence of recharging stations, 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 above three models are depicted in Fig. 9. The total number of trainable parameters and the training time for one episode are summarized in Table1. Simulation results in terms of the basic Travel Sales Problems (TSPs) and the concerned DDP-Rs are shown in Table 2.

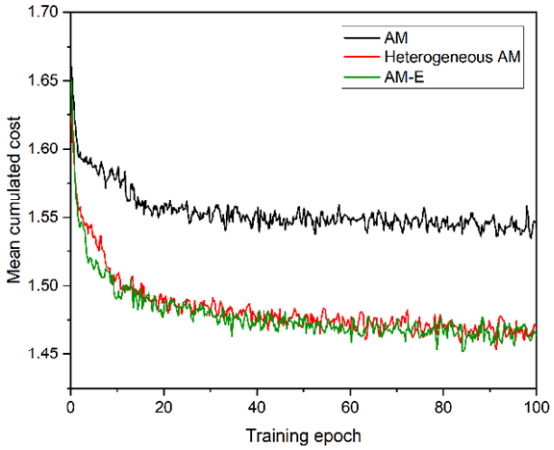

Fig. 9. Learning curves for solving the DDP-R with 10 delivery requests and 3 recharge stations.

Table 1 Total trainable parameters of models and running time for one epoch. Programs all ran in Tesla V100. Time is measured over the entire training set and averaged.

	AM model	Heterogeneous AM model	AM-E model
Trainable parameters	693632	1086848	746112
Running time	4min50s	8min18s	6min07s

Table 2 Comparison among AM, AM-E and heterogeneous AM models, results indicated with * are reported by [19].

Task	Optimal	AM model	Heterogeneous AM model	AM-E model
TSP20	3.83*	3.84 (0.33%)*	3.84 (0.33%)*	3.84 (0.33%)
TSP50	5.69*	5.82 (2.28%)*	5.82 (2.28%)*	5.81(2.11%)
TSP100	7.76*	8.19 (5.49%)*	8.19 (5.49%)*	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%)

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 AM in solving DDP-Rs, while no 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, so we refer to the same results in the table.

For further comparison, the proposed AM-E model achieves comparable and even slightly better performance than the heterogeneous attention model with about a 32% reduction in parameters. In the heterogeneous attention model, 8 extra types of attention layers are added to the original one for solving DDP-Rs. Even though parameters of all *keys* and *values* are shared, the *query* matrices of attention layers are kept independently, which introduces a large number of extra trainable parameters to the model and almost doubles the running time for one episode. More trainable parameters always imply more computational amount and larger memory requirement. In the proposed AM-E model,

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 model for solving deterministic DDP-R without considering the wind. The experiment is carried out with 3 different dimensions (the number of delivery requests □ □ □□□ □□□BCE □□□ while the number of charging stations is fixed to 3) and compared to the original AM model and deterministic baseline methods.

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

Regarding the DRL implmentations, we introduce the original AM model as another baseline. For the heterogeneous AM model, since the proposed AM-E model outperforms it in terms of both solution qualities and computing efficiencies as demonstrated by simulations in the last section, so we remove the hetegenerous AM model from the following comparison studies. With regard to the proposed AM-E model and the baseline AM model, we apply two decoding strategies for evaluation: 1) *Greedy*, always select the action with maximum probability at each step; 2) *Sampling*, sample □ □ □□□□ routes for each instance according to the probability distribution and choose the one with minimum cost. The simulation results are summarized in Table 3.

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

Method	DDP10-R3			DDP20-R3			DDP40-R3		
	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 (*1s*)	1.493 ± 0.036	0.81%	1.0	3.438 ± 0.034	27.90%	1.0	-	-	-
Or-tools(*10s*)	**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(*50s*)	-	-	-	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(*Sample1280*)	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*)	1.526 ± 0.005	3.04%	0.063	2.790 ± 0.006	3.79%	0.121	5.149 ± 0.007	2.18%	0.416
AM-E (*Sample1280*)	**1.487 ± 0.004**	**0.41%**	**0.131**	**2.709 ± 0.005**	**0.79%**	**0.286**	**5.039 ± 0.007**	**0.00%**	**0.465**

From Table 3, Gurobi fails to obtain a feasible solution in some instances within 100 seconds, 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 since 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 model shows a performance improvement over its basic form, the AM model, both for greedy decoding and sampling decoding. Overall, AM-E has better scalability, and 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 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 □□ □□□ will be imposed on the overall cost. The approaches are tested in a stochastic environment with the mean wind value □□ □□ c□ Since 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.

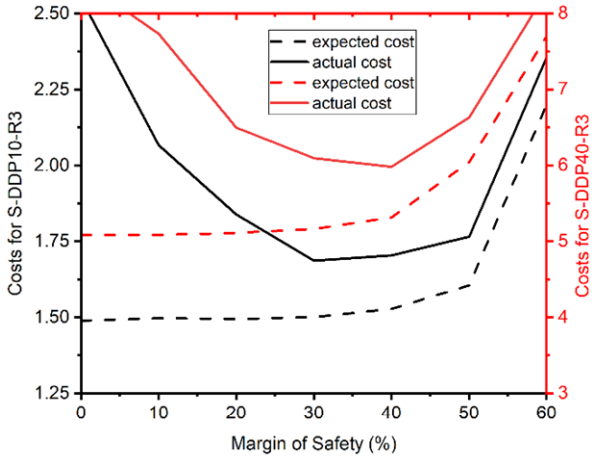

Fig. 10. Expected costs and actual costs with different margins of safety

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

Method	S-DDP10-R3			S-DDP20-R3			S-DDP40-R3		
	Cost	Gap	Time(s)	Cost	Gap	Time(s)	Cost	Gap	Time(s)
Gurobi (*margin 20%*)	1.783 ± 0.816 (99.90%)	5.69%	0.416	3.161 ± 1.185 (97.80%)	1.38%	8.354	6.423 ± 2.674 (41%)	-	48.846
Gurobi (*margin 30%*)	1.716 ± 0.600 (99.89%)	1.72%	0.503	3.145 ± 1.138 (98.30%)	0.87%	11.239	6.209 ± 2.418 (52%)	-	71.887
Or-tools (*margin 20%*)	1.839 ± 0.672	9.01%	1.0	3.418 ± 1.581	9.62%	10.0	6.495 ± 3.431	11.98%	50.0
Or-tools (*margin 30%*)	**1.687 ± 0.300**	**0.00%**	**1.0**	3.341 ± 1.255	7.15%	10.0	6.097 ± 3.045	5.12%	50.0
AM(*Greedy*)	1.781 ± 0.007	5.57%	0.102	3.234 ± 0.012	3.72%	0.192	5.937 ± 0.019	2.36%	0.350
AM-E(*Greedy*)	1.744 ± 0.009	3.38%	0.071	**3.118 ± 0.011**	**0.00%**	**0.133**	**5.800 ± 0.019**	**0.00%**	**0.343**

Table 4 compares the proposed AM-E model, AM model and offline planning baselines with the 20% and 30% margin 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 while DRL methods perform better with scenarios of 20 requests 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. However, how to define the margin value while operating in varying wind fields is a tricky question. It might either become risky or conservative. As an alternative solution, the DRL approaches to avoid this problem by continuously adjusting the route according to the UAV's remaining energy.

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, AM-E offers better solutions than AM without consuming more time. Moreover, since the DRL-based method possesses the ability to respond to environmental changes, it is expected to perform better in dynamic scenarios compared to the deterministic approaches which require replanning of the whole routes.

VI.Conclusion and Future Work

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, we have built a simulated environment 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 model has been presented and demonstrated with high-quality solutions. AM-E model 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 model has been trained and evaluated via the simulated environment. Results showed the AM-E is fast, robust and has better scalability compared to three baseline heuristic methods and a state-of-the-art deep learning model.

The extension to multi-drone 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 tends 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 inter-drone 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. We plan to collect more realistic data from UAV delivery operations for validation. Building a more refined simulation environment that considers time windows, quality of service, and dynamic events are included in future research plans.

Appendix

A. *Mixed Integer Programming* model for DDP-Rs

With the objective to minimize the energy cost, the concerned DDP-Rs claimed in Section 3.1, are formulated into *Mixed Integer Programming* (MIP). The indices, sets, parameters and decision variables in the MIP model are defined as follows:

Table A.1 the notations used in the MIP model

Indices and Sets	
i, j	indices of nodes
P/D	set of pickup/delivery nodes, $\mathbb{P} = \{1,2,\dots,n\}$, $\mathbb{D} = \{n+1, n+2, \dots, 2n\}$
R	set of recharging stations
O	set of the depot node and its one copy, $\mathbb{O} = \{n_0, n_0'\}$
N	set of all nodes, $\mathbb{N} = \mathbb{O} \cup \mathbb{P} \cup \mathbb{D} \cup \mathbb{C}$
Decision parameters	
x_{ij}	1 if vehicle travels from node i to node j; 0 otherwise
y_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}	electric energy (kWh) needed to travel from node i to node j
y_{min}	lower bound of SoC constraints (%)
y_{max}	upper bound of SoC constraints (%)
B_{max}	maximum battery capacity (kWh)
τ_{ij}	travel time from node i to node j
M	a big positive value

The proposed MIP model is as follows:

$$\min \sum_{i \in N} \sum_{j \in N} e_{ij} x_{ij}$$

Subject to:

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 [kWh]	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	$C_{D_{body}}$	0.3
Blade drag	c_{bd}	0.075
Rotor mean chord	$\bar{c}$	0.1
Blade lift	$\overline{c_l}$	0.4
Lifting power markup	κ_{ind}	1.15

Acknowledgments

An Acknowledgments section, if used, **immediately precedes** the References. Sponsorship information and funding data are included here. The preferred spelling of the word "acknowledgment" in American English is without the "e" after the "g." Avoid expressions such as "One of us (S.B.A.) would like to thank…" Instead, write "F. A. Author thanks…"

The authors

References

[1] Lin, J., Zhou, W. and Wolfson, O. (2016) 'Electric Vehicle Routing Problem', *Transportation Research Procedia*. Elsevier B.V., 12(June 2015), pp. 508–521. doi: 10.1016/j.trpro.2016.02.007.

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

[3] Erdelic, T., Carić, T. and Lalla-Ruiz, E. (2019) 'A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches', *Journal of Advanced Transportation*, 2019, pp. 1–27. doi: 10.1155/2019/5075671.

[4] Toth, P. and Daniele, V. (2002) 'The Vehicle Routing Problem', *Society for Industrial and Applied Mathematics*, 9, pp. 175–186. doi: 10.1007/978-0-387-77610-1_13.

[5] Tahami, H., Rabadi, G. and Haouari, M. (2020) 'Exact approaches for routing capacitated electric vehicles', *Transportation Research Part E: Logistics and Transportation Review*. Elsevier Ltd, 144(November), p. 102126. doi: 10.1016/j.tre.2020.102126.

[6] Desaulniers, G., Errico, F., Irnich, S. and Schneider, M. (2016) 'Exact algorithms for electric vehicle-routing problems with time windows', *Operations Research*, 64(6), pp. 1388–1405. doi: 10.1287/opre.2016.1535.

[7] Kucukoglu, I., Dewil, R. and Cattrysse, D. (2021) 'The electric vehicle routing problem and its variations: A literature review', *Computers and Industrial Engineering*. Elsevier Ltd, 161(July), p. 107650. doi: 10.1016/j.cie.2021.107650.

[8] Ruland, K. S. and Rodin, E. Y. (1997) 'The pickup and delivery problem: Faces and branch-and-cut algorithm', *Computers and Mathematics with Applications*, 33(12), pp. 1–13. doi: 10.1016/S0898-1221(97)00090-4.

[9] Ropke, S. and Cordeau, J. F. (2009) 'Branch and cut and price for the pickup and delivery problem with time windows', *Transportation Science*, 43(3), pp. 267–286. doi: 10.1287/trsc.1090.0272.

[10] Chiang, W. C. and Russell, R. A. (1996) 'Simulated annealing metaheuristics for the vehicle routing problem with time windows', *Annals of Operations Research*, 63, pp. 3–27. doi: 10.1007/BF02601637.

[11] Homberger, J. and Gehring, H. (1999) 'Two evolutionary metaheuristics for the vehicle routing problem with time

windows', *INFOR Journal*, 37 (O)(3), pp. 297–318. doi: 10.1080/03155986.1999.11732386.

[12] de Oliveira da Costa, P. R., Mauceri, S., Carroll, P. and Pallonetto, F. (2018) 'A Genetic Algorithm for a Green Vehicle Routing Problem', *Electronic Notes in Discrete Mathematics*. Elsevier B.V., 64, pp. 65–74. doi: 10.1016/j.endm.2018.01.008.

[13] Rastani, S. and Çatay, B. (2021) 'A large neighborhood search-based matheuristic for the load-dependent electric vehicle routing problem with time windows', *Annals of Operations Research*. Springer US. doi: 10.1007/s10479-021-04320-9.

[14] Ghilas, V., Demir, E. and Van Woensel, T. (2016) 'An adaptive large neighborhood search heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines', *Computers and Operations Research*. Elsevier, 72, pp. 12–30. doi: 10.1016/j.cor.2016.01.018.

[15] Goeke, D. (2019) 'Granular tabu search for the pickup and delivery problem with time windows and electric vehicles', *European Journal of Operational Research*. Elsevier B.V., 278(3), pp. 821–836. doi: 10.1016/j.ejor.2019.05.010.

[16] Vinyals, O., Brain, G., Fortunato, M., Jaitly, N. and Brain, G. (no date p) 'Pointer Networks', pp. 1–9.

[17] Bello, I., Pham, H., Le, Q. V., Norouzi, M. and Bengio, S. (2019) 'Neural combinatorial optimization with reinforcement learning', *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, pp. 1–15.

[18] Nazari, M., Oroojlooy, A., Takáč, M. and Snyder, L. V. (2018) 'Reinforcement learning for solving the vehicle routing problem', *Advances in Neural Information Processing Systems*, 2018-Decem, pp. 9839–9849.

[19] Kool, W., Van Hoof, H. and Welling, M. (2018) 'Attention, learn to solve routing problems!', *arXiv*.

[20] Bono, G., Dibangoye, J. S., Simonin, O., Matignon, L. and Pereyron, F. (2020) 'Solving Multi-Agent Routing Problems Using Deep Attention Mechanisms', *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10. doi: 10.1109/tits.2020.3009289.

[21] Yu, J. J. Q., Yu, W. and Gu, J. (2019) 'Online Vehicle Routing with Neural Combinatorial Optimization and Deep Reinforcement Learning', *IEEE Transactions on Intelligent Transportation Systems*. IEEE, 20(10), pp. 3806–3817. doi: 10.1109/TITS.2019.2909109.

[22] Lin, B., Ghaddar, B. and Nathwani, J. (2020) 'Deep reinforcement learning for electric vehicle routing problem with time windows', *arXiv*, pp. 539–547.

[23] Li, J., Xin, L., Cao, Z., Lim, A., Song, W. and Zhang, J. (2021) 'Heterogeneous Attentions for Solving Pickup and Delivery Problem via Deep Reinforcement Learning', *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10. doi: 10.1109/TITS.2021.3056120.

[24] Bono, G. (2021) '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'.

[25] Basso, R., Kulcsár, B. and Sanchez-Diaz, I. (2021) 'Electric vehicle routing problem with machine learning for energy prediction', *Transportation Research Part B: Methodological*. Elsevier Ltd, 145, pp. 24–55. doi: 10.1016/j.trb.2020.12.007.

[26] Kirschstein, T. (2020) 'Comparison of energy demands of drone-based and ground-based parcel delivery services', *Transportation Research Part D: Transport and Environment*. Elsevier, 78(December 2019), p. 102209. doi: 10.1016/j.trd.2019.102209.

[27] Langelaan, J. W., Schmitz, S., Palacios, J. and Lorenz, R. D. (2017) 'Energetics of rotary-wing exploration of Titan', *IEEE Aerospace Conference Proceedings*. IEEE. doi: 10.1109/AERO.2017.7943650.

[28] Johnson, G. L. (1985) *Wind Energy Systems*, *Prentice-Hall*. Englewood cliffs. doi: 10.1109/JPROC.2017.2695485.

[29] Moorhouse, D. J. and Woodcock, R. J. (1982) 'Background Information and User Guide for MIL-F-8785B, Military Specification - Flying Qualities of Piloted Airplanes', *Mil-F-8785C*, p. 255. Available at: http://www.dept.aoe.vt.edu/~durham/AOE5214/MILSPEC8785C.pdf.

[30] Gong, L. and Cheng, Q. (2019) 'Exploiting edge features for graph neural networks', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June, pp. 9203–9211. doi: 10.1109/CVPR.2019.00943.

[31] Wang, Z., Chen, J. and Chen, H. (2021) 'EGAT: Edge-Featured Graph Attention Network', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12891 LNCS, pp. 253–264. doi: 10.1007/978-3-030-86362-3_21.

[32] Hussain, M. S., Zaki, M. J. and Subramanian, D. (2021) 'Edge-augmented Graph Transformers: Global Self-attention is Enough for Graphs', pp. 1–19. Available at: http://arxiv.org/abs/2108.03348.

[33] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017) 'Attention is all you need', *arXiv*.