1-2024

DL-DRL: A double-level deep reinforcement learning approach for large-scale task scheduling of multi-UAV

Xiao MAO

Guohua WU

Mingfeng FAN

Zhiguang CAO
Singapore Management University, zgcao@smu.edu.sg

Witold PEDRYCZ

DL-DRL: A double-level deep reinforcement learning approach for large-scale task scheduling of multi-UAV

Xiao Mao, Zhiguang Cao, Mingfeng Fan, Guohua Wu*, and Witold Pedrycz, *Life Fellow, IEEE*

***Abstract*—Exploiting unmanned aerial vehicles (UAVs) to execute tasks is gaining growing popularity recently. To solve the underlying task scheduling problem, the deep reinforcement learning (DRL) based methods demonstrate notable advantage over the conventional heuristics as they rely less on hand-engineered rules. However, their decision space will become prohibitively huge as the problem scales up, thus deteriorating the computation efficiency. To alleviate this issue, we propose a double-level deep reinforcement learning (DL-DRL) approach based on a divide and conquer framework (DCF), where we decompose the task scheduling of multi-UAV into task allocation and route planning. Particularly, we design an encoder-decoder structured policy network in our upper-level DRL model to allocate the tasks to different UAVs, and we exploit another attention based policy network in our lower-level DRL model to construct the route for each UAV, with the objective to maximize the number of executed tasks given the maximum flight distance of the UAV. To effectively train the two models, we design an interactive training strategy (ITS), which includes pre-training, intensive training and alternate training. Experimental results show that our DL-DRL performs favorably against the learning-based and conventional baselines including the OR-Tools, in terms of solution quality and computation efficiency. We also verify the generalization performance of our approach by applying it to larger sizes of up to 1000 tasks. Moreover, we also show via an ablation study that our ITS can help achieve a balance between the performance and training efficiency. Our code is publicly available [1].**

***Index Terms*—Deep reinforcement learning, divide and conquer-based framework, interactive training, multi-UAV task scheduling.**

I. Introduction

Nowadays, unmanned aerial vehicles (UAVs) have been widely applied in the areas of package delivery [1], environment surveillance [2], target tracking [3], and reconnaissance [4] due to their high flexibility, strong mobility, and low power consumption. As the missions to be executed become more and more complex in terms of scale and difficulty, the multi-UAV system has been recognized with salient superiority to the single UAV, where task scheduling plays a vital role to enable the cooperation among the UAVs.

As a variant of the multiple traveling salesman problem (mTSP) [5], the multi-UAV task scheduling problem is known to be NP-hard [6], for which it is difficult to find an optimal solution in a short computation time. To solve this problem, both exact and heuristic algorithms have been investigated and explored. Exact algorithms [7]–[9] can deliver optimal solutions on small-scale problems, whereas they are relatively incapable on large-scale ones due to the exponentially increasing computation time. Heuristic algorithms, such as genetic algorithm (GA) [10], ant colony optimization (ACO) [11], particle swarm optimization (PSO) [12], and simulated annealing algorithm (SA) [13] are desirable to tackle the large-scale problems, which strike a balance between solution quality and computation efficiency. However, the rules in heuristic algorithms always need to be manually designed with domain knowledge and efforts, which may still limit the eventual performance. Additionally, previous studies for the multi-UAV task scheduling usually solve the problem in a whole manner, i.e., directly determining the execution order (or route) of tasks for each UAV, which inevitably runs into the "curse of dimensionality" when solving large-scale problems.

To tackle the large-scale task scheduling for multi-UAV, a number of recent studies decompose the original problem into several subproblems, which are then solved by conventional heuristics [14], [15]. In this way, the computation complexity could be effectively reduced, thus making the decomposition scheme more favorable for handling large-scale problems. However, the hand-crafted heuristics in those methods may still hold back the performance, as they fail to leverage the underlying pattern among the problem instances to improve the overall performance. On the other hand, with the advances of deep learning (DL) and reinforcement learning (RL), deep reinforcement learning (DRL) has been widely explored in games [16], [17], robotics [18], and natural language processing [19]. Recently, DRL is also intensively applied in combinatorial optimization problems (COPs), such as capacitated vehicle routing problem (CVRP) [20], traveling salesman problem (TSP) [21], and orienteering problem (OP) [22]. In contrast to the conventional heuristics with hand-crafted rules, DRL is able to automatically learn a decision-making policy in a data-driven manner by leveraging the underlying pattern among the COP instances. However, when the problem scales

* Corresponding Author.

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

Zhiguang Cao is with the School of Computing and Information Systems, Singapore Management University, Singapore. (E-mail: zhiguangcao@outlook.com).

Witold Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada, also with the Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia, and also with the Systems Research Institute, Polish Academy of Sciences, 01447 Warsaw, Poland. (e-mail: wpedrycz@ualberta.ca).

[1]https://faculty.csu.edu.cn/guohuawu/zh_CN/zdylm/193832/list/index.htm

up, the decision space in DRL also expands sharply, which may cause unstable training and undesirable performance [23].

To alleviate the issue, in this paper, we exploit a divide and conquer framework (DCF) to decompose the original multi-UAV scheduling problem into a task allocation subproblem and a route planning subproblem. Then, given the decomposed decision space, we propose a double-level deep reinforcement learning approach (DL-DRL) to solve the two subproblems, respectively. In particular, the upper-level DRL model is mainly responsible for the task allocation, in which an encoder based on the self-attention mechanism [24] and a UAV selection decoder are designed. The lower-level DRL model shared by all UAVs is mainly responsible for the route planning, where an encoder-decoder structure inspired by the well-known attention model (AM) [25] is designed. Unlike the general hierarchical DRL which needs to learn the goal/option determination by trial and error [26], [27], the goals of our DL-DRL are determined by the DCF in advance. In this way, the policy learning is reduced to only the actions selection in both levels. Furthermore, we also propose an interactive training strategy (ITS) to train the two DRL models given the potential interplay between them, which includes the pre-training, intensive training, and alternate training. Experimental results justified the effectiveness of our approach in achieving competitive solution quality and computation efficiency, and its desirable generalization to larger problem instances. The main contributions of this paper are summarized as follows.

- A divide and conquer framework (DCF) is exploited to decompose the multi-UAV task scheduling problem into the task allocation and route planning subproblems. Based on the decomposition, a double-level DRL approach is proposed, in which the encoder-decoder structured policy networks are developed in both the upper-level and lower-level DRL models to solve the two different subproblems, respectively.
- An interactive training strategy (ITS) is proposed to train the upper-level and lower-level DRL models. This strategy comprises three procedures, including pre-training, intensive training, and alternate training, which could attain desirable balance between training performance and efficiency.
- Extensive experiments on various scenarios suggest that our DL-DRL approach is able to achieve favorable solution quality and computation efficiency against the learning-based and conventional baselines. The efficacy of our training strategy is also justified through an ablation. Additionally, by applying the model learned for a problem size to solve larger ones (up to 1000 tasks), our DL-DRL approach is verified the desirable generalization performance.

The remainder of this paper is organized as follows. Section II briefly reviews the related works on the multi-UAV task scheduling problem. Section III presents the mathematical formulation of the problem and our divide and conquer framework for problem decomposition. Section IV elaborates our double-level DRL approach and iterative training strategy. Section V conducts extensive experiments and results analysis. Section VI concludes the paper.

II. Related Work

Similar to m-TSP, the multi-UAV task scheduling problem is also a kind of COP with additional constraints like the maximum travelling distance due to the limited battery. Typically, most of the works build a mixed-integer linear program model [7] for this problem and design exact or heuristic algorithms to solve it.

Exact algorithms such as branch and bound [28], branch and price [29], column generation [30], and dynamic programming [8] have the potential to attain optimal solutions, while they are unsuitable for solving large-scale problems due to the prohibitive computation time for exhaustive search. Thus, heuristic algorithms that could reduce search space with carefully designed heuristic rules are reckoned as desirable alternatives for solving the multi-UAV task scheduling problem. Ye et al. [31] developed an GA method with a multi-type gene chromosome encoding scheme and an adaptive operation for efficient solution searching. Shang et al. [32] proposed a hybrid algorithm by combining GA and ACO, in which the inferior individuals of the population in GA are replaced with the superior ones in ACO during the population evolution. Chen et al. [33] proposed a modified two-part wolf pack search (MTWPS) algorithm based on a designed transposition and extension operation to solve the problem. However, the hand-crafted rules highly rely on the human experience and domain knowledge, hence the solution quality may be undermined.

DRL which takes the advantages of both DL and RL is wildly investigated across various areas like games, robotics, speech recognition, and natural language processing [34] due to its powerful learning capability. As DRL can automatically learn a decision-making policy via the interaction with the environment, it has been recently applied to solve COPs [35], [36], which learns the rules from a large number of problem instances rather than manually designing them. Bello et al. [37] presented a DRL model based on pointer networks (PtrNet) [38] to solve the TSP, which inspired many subsequent works for DRL to solve COPs. To improve the performance of PtrNet-based DRL, Ma et al. [39] proposed a graph pointer network that integrates the graph neural network and PtrNet for better feature extraction. In addition, Kool et al. [25] proposed the AM (Attention Model) method based on the Transformer architecture [24] to solve a variety of COPs, which outperforms a wide range of learning-based and conventional baselines. Xu et al. [40] improved the AM to solve VRPs more efficiently by incorporating a multiple relational attention mechanism. These DRL methods exhibit strong performance and reasonable computation efficiency for solving COPs, whereas the training becomes unstable or even out of reach for large-scale problems due to the huge decision space [23].

In summary, due to the explosively increasing computational overhead and hand-engineered rules, conventional exact and heuristic algorithms are frustrated in solving the large-scale task scheduling problem of multi-UAV. DRL which automatically learns a decision-making policy through the interaction

with the environment (or problem instances) tends to be a desirable alternative. However, the considerable expansion of decision space also hinders the DRL model from training on large-scale problems. To tackle this issue, recent works have concentrated on the problem decomposition. Hu et al. [41] divided the M-TSP into several small-scale TSP via the proposed DRL policy networks. Liu et al. [42] constructed a decomposition framework named EA-DRL to decompose the OP into a knapsack problem and a TSP which are solved by integrating an evolutionary algorithm and DRL, respectively. In this way, the large-scale problem is divided into several small subproblems that DRL can handle efficiently given the relatively small decision space. Inspired by them, we design a DCF that decomposes the large-scale multi-UAV task scheduling problem into two subproblems, i.e., task allocation and UAV route planning. With the DCF, we also propose a double-level DRL approach to solve the two subproblems in the upper level and the lower level, respectively, which has favorable potential for solving the large-scale multi-UAV task scheduling problems.

III. Problem Formulation and Framwork Description

In this section, a mixed-integer linear program model is presented for the multi-UAV task scheduling problem, where the objective is to maximize the total number of tasks completed by UAVs given their maximum flight distance. To solve the large-scale problem, we decompose the original problem into a task allocation subproblem and a UAV route planning subproblem based on the divide and conquer framework (DCF), where the two subproblems are modelled in the upper level and lower level of the DCF, respectively.

A. Mathematical Formulation

Let $T = \{0, 1, \cdots, N\}$ be the task set, where 0 and N denote the UAV depot, which a UAV must depart from and finally return to after completing the assigned tasks. Let $U = \{1, 2, \cdots, V\}$ be the UAV set, where V is the number of UAVs. In this paper, all UAVs are considered as homogeneous ones with a same maximum flight distance D. Three types of decision variables are defined in the mathematical model of the multi-UAV task scheduling problem, i.e., 1) x_{ijk} is a binary variable that equals 1 if the UAV $k \in U$ completes the task j right after the task i and 0 otherwise ($i, j \in T$); 2) y_{ik} is also a binary variable that equals 1 if the UAV $k \in U$ has executed the task $i \in T$ and 0 otherwise; 3) z_{ik} is a continuous variable indicating the allowed remaining flight distance of the UAV $k \in U$ when visiting task $i \in T$. Accordingly, the mathematical model is formulated as follows,

$$\max \sum_{i=1}^{N-1} \sum_{k=1}^{V} y_{ik} \tag{1}$$

Subject to:

$$\sum_{k=1}^{V} y_{ik} \leq 1, i \in \{1, 2, \cdots, N-1\} \tag{2}$$

$$\sum_{j=1}^{N} x_{ijk} = y_{ik}, i \in \{1, 2, \cdots, N-1\}, k \in U \tag{3}$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=0}^{N-1} x_{jik} \leq 1, i \in \{0, 1, \cdots, N\}, k \in U \tag{4}$$

$$\sum_{k=1}^{V} \sum_{j=1}^{N-1} x_{0jk} = \sum_{k=1}^{V} \sum_{i=1}^{N-1} x_{iNk} = V \tag{5}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} d_{ij} \cdot x_{ijk} \leq D, k \in U \tag{6}$$

$$z_{ik} - z_{jk} + D \cdot x_{ijk} \leq D - d_{ij}, i \neq j \in T, k \in U \tag{7}$$

$$0 \leq z_{ik} \leq D - d_{i0}, i \in \{1, 2, \cdots, N\}, k \in U \tag{8}$$

where the objective function (1) aims to maximize the total number of completed or executed tasks; Constraints (2) to (4) ensure that each task can be completed once at most; Constraint (5) restricts that all the UAVs must depart from and return to the depot; Constraint (6) ensures the compliance of the maximum flight distance for each UAV; Subtours are eliminated with constraints (7) and (8).

B. Divide and Conquer Framework

To reduce the computational complexity of the large-scale multi-UAV task scheduling problem, we decompose it into two subproblems, i.e., task allocation and UAV route planning, based on the divide and conquer framework (DCF). As illustrated in Fig. 1, the DCF comprises an upper level and a lower level. Given a problem instance, at the beginning, the tasks are allocated to different UAVs in the upper level. Afterwards, each UAV performs the route planning for the allocated tasks in the lower level. However, during the route planning, some tasks are possibly to be excluded out due to the maximum flight distance of the UAV.

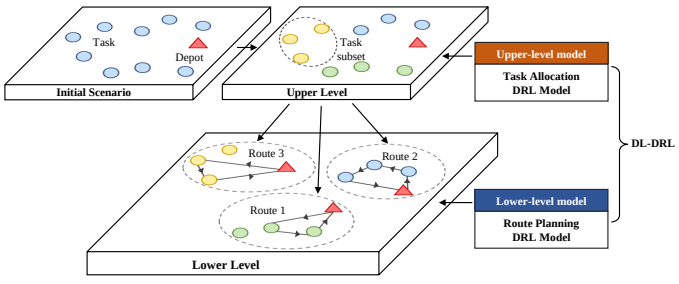


Fig. 1: Divide and conquer framework for task scheduling.

Based on this DCF, we propose a DL-DRL approach to solve the multi-UAV task scheduling problem, in which two different DRL models are build in the upper level and lower level, respectively. In the upper level, we design a task allocation DRL model to select an appropriate UAV for each given task. In the lower level, we design a route planning DRL model to construct the route for each UAV, with the objective to maximize the number of tasks to be executed given the maximum flight distance of the UAV. In our DL-DRL, the two models interact with each other, i.e., the upper-level model feeds the allocated tasks as the input to the lower-level

model, while the reward of the upper-level model relies on the output of the lower-level model. Through the combination of the models in the two levels, the DL-DRL approach is able to solve the multi-UAV task scheduling problem more efficiently. Moreover, our DL-DRL also has the potential to solve other similar COPs, such as m-TSP [43], OP [44], etc.

IV. Methodology

In this section, we introduce our DL-DRL approach for solving the multi-UAV task scheduling problem. For the task allocation subproblem in the upper level, we first cast it as an Markov decision process (MDP) and then design a policy network for the task allocation DRL model. For the UAV route planning subproblem in the lower level, we also design a policy network to construct the route that maximizes the number of tasks to be executed by the UAV given its maximum flight distance. Note that all the UAVs share the same route construction policy in our approach. Afterwards, we propose an interactive training strategy (ITS) that consists of pre-training, intensive training, and alternate training to achieve a balance between the performance and training efficiency.

A. Upper-level DRL Model for Task Allocation

1) Markov Decision Process: The procedure of the task allocation in the upper level can be deemed as a sequential decision-making process, where a UAV will be assigned for each given task. We cast such a process as an MDP which includes state space S, action space A, state transition rule P, and reward R. The detailed definition of our MDP is stated as follows.

State. The state includes the current information of the UAV and task at each step, i.e., $s_t = (V_t, x_t) \in S$, where V_t is the task set that have been allocated to UAVs at time step t, and x_t is the task that need to be allocated at time step t and represented using its coordinates. Particularly, $V_t = \left\{U_t^1, U_t^2, \cdots, U_t^v\right\} = \left\{\{u_1^1, u_2^1, \cdots, u_t^1\}, \{u_1^2, u_2^2, \cdots, u_t^2\}, \cdots, \{u_1^v, u_2^v, \cdots, u_t^v\}\right\}$, where U_t^i and u_t^i are the task set and the corresponding task features (i.e., coordinates of the task) of the UAV i at time step t, respectively.

Action. The action is to allocate the current given task to a UAV, which could be expressed as $a_t = (u_i, x_t) \in A$, i.e., the task x_t will be allocated to the UAV i at time step t.

State transition rule. The state transition rule indicates that the state s_t is transited to s_{t+1} after taking the action a_t, i.e., $s_{t+1} = (V_{t+1}, x_{t+1}) = P(s_t, a_t)$. The x_{t+1} refers to the next task according to the order of the task sequence in the input instance. The elements in V_t are updated as follows,

$$U_{t+1}^j = \begin{cases} \left[U_t^j, x_t^i\right], j = i \\ \left[U_t^j, u_t^j\right], otherwise \end{cases} \tag{9}$$

where $[,]$ refers to the concatenation operation, x_t^i is the task allocated to UAV i at time step t, and u_t^j is the last element in U_t^j (i.e., the task allocated to UAV j at time step t-1). We use this update strategy to keep the same dimension of the allocated task set for each UAV, which may cause repeated tasks for a UAV but would be much convenient for the subsequent batch training.

Reward. Since we aim to maximize the number of tasks that could be executed, we define the reward as the total number of tasks completed by all UAVs, i.e., $R = \sum_{i=1}^{v} T_i$, where T_i is the number of tasks completed by UAV i, which is obtained based on the route planning DRL model in the lower level.

2) Architecture of the Policy Network: Compared to the conventional Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, the Transformer has stronger capability in handling sequential data, and has been widely employed in domains of natural language processing [45], image recognition [46], recommendation system [47], and combinatorial optimization [40] as well. Regarding the task allocation, we propose a Transformer-style policy network which is concretized by an encoder and a decoder to allocate a given task to the appropriate UAV, as illustrated in Fig. 2.

Encoder. The inputs of the encoder are the raw features of tasks and the depot, say the coordinates, and we exploit a linear projection layer and multiple attention layers to learn more informative embeddings. Specifically, the raw inputs are first encoded to d_h-dimensional node embeddings h^0 via the linear projection with d_h=128. Then the node embeddings h^0 are promoted to h^L through L attention layers, each of which consists of a multi-head attention (MHA) sublayer and a feed-forward (FF) sublayer.

In the l-th attention layer, the input $h^{l-1} = \left\{h_1^{l-1}, h_2^{l-1}, \ldots, h_N^{l-1}\right\}$ is the output of the last attention layer, where N is the number of tasks. To attain the output of the l-th attention layer h^l, the MHA sublayer processes $h^{l\text{-}1}$ via the multi-head self-attention mechanism first. In specific, for each head, the vectors of $query$, key, and $value$ are derived by multiplying $h^{l\text{-}1}$ and the corresponding trainable parameters. Then, we calculate the attention value Y_{lm} according to Eq. (11), where $m = \{1, 2, \cdots, M\}$ with M=8 and $dim = \frac{128}{M}$. After the calculation of attention values, we concatenate the attention values of all heads and project it to a d_h-dimensional vector, and then yield the node embeddings $\hat{h}^l$ through the skip-connection and batch normalization operations. Afterwards, $\hat{h}^l$ is fed to the FF sublayer which is concretized by two linear projections and a ReLu activation function. Here, the skip-connection and batch normalization are also used for the output of the FF sublayer so as to derive the eventual output of the l-th attention layer. In general, this process could be formulated as follows,

$$q_{lm} = W_{lm}^Q h^{l-1}, k_{lm} = W_{lm}^K h^{l-1}, v_{lm} = W_{lm}^V h^{l-1}, \tag{10}$$

$$Y_{lm} = \text{softmax}\left(\frac{q_{lm}^T k_{lm}}{\sqrt{dim}}\right) v_{lm}, \tag{11}$$

$$\text{MHA}\left(h^{l-1}\right) = \left[Y_{l1}; Y_{l2}; \ldots; Y_{lM}\right] W_l^O, \tag{12}$$

$$\hat{h}^l = BN^l\left(h^{l-1} + \text{MHA}\left(h^{l-1}\right)\right), \tag{13}$$

$$h^l = BN^l\left(\hat{h}^l + \text{FF}\left(\hat{h}^l\right)\right), \tag{14}$$

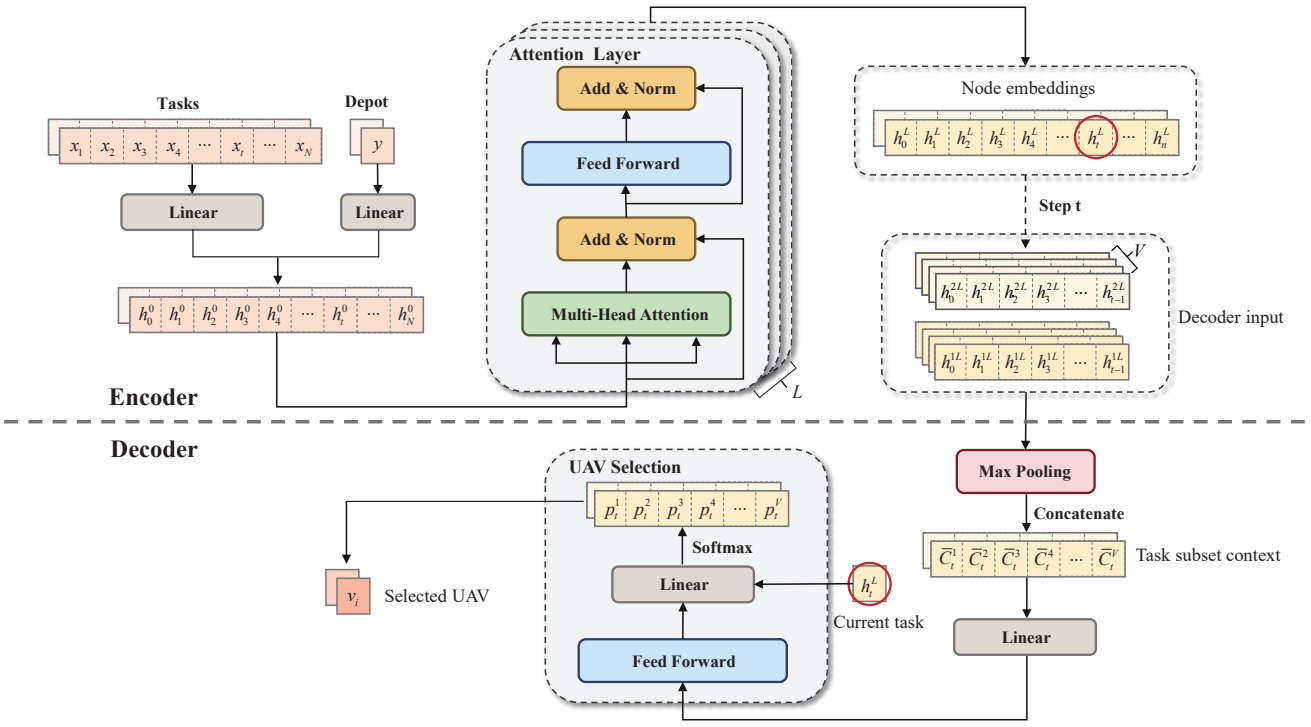


Fig. 2: Architecture of policy network for task allocation in the upper-level DRL model.

where $W_{lm}^Q, W_{lm}^K, W_{lm}^V \in \mathbb{R}^{M\times 128\times dim}$ and $W_l^O \in \mathbb{R}^{128\times 128}$ are trainable parameters in the l-th attention layer.

Decoder. The output of the encoder h^L is used as the input of the decoder. In the decoding process, we choose a task according to the order in the input instance, and then select an appropriate UAV for this task at the current time step and add the task information (i.e., node embedding) to the task subset of the UAV. Specifically, we construct an allocation feature context for UAV i, i.e., $\tilde{C}_t^i = \left[h_0^{iL}, h_1^{iL}, \ldots, h_j^{iL}, ..., h_{t-1}^{iL}\right]$, where h_j^{iL} is the node embedding of the task that is allocated to UAV i at the time step j. We aggregate all the allocation feature contexts via a max pooling and concatenate them to obtain a task subset context $\hat{C}_t^{TS}$. After that, a linear projection and a 512-dimensional FF layer are used to yield the task subset embeddings H_t^{TS}. Finally, we concatenate H_t^{TS} and node embedding of the current task h_t^L, and then feed it to a linear projection and the softmax function to attain the output probability for task allocation. Based on the probability, we select a UAV for the given task using either greedy decoding or sampling decoding. The formmer always selects the UAV with the highest probability, while the latter selects the UAV by sampling according to its probability. The overall decoding process could be briefly expressed as follows,

$$\bar{C}_t^i = \max\left(\tilde{C}_t^i\right), \tag{15}$$

$$\hat{C}_t^{TS} = \left[\bar{C}_t^1, \bar{C}_t^2, \ldots, \bar{C}_t^V\right], \tag{16}$$

$$H_t^{TS} = FF(W_1 \hat{C}_t^{TS} + b_1), \tag{17}$$

$$p_t = \text{softmax}(W_2\left[H_t^{TS}, h_t^N\right] + b_2), \tag{18}$$

where W_1, b_1, W_2 and b_2 are trainable parameters.

B. Lower-level DRL Model for Route Planning

Given the tasks allocated by the upper-level model, we seek to construct a viable route for each UAV to execute those tasks in the lower level, where we design another DRL model to achieve this goal. In light of the nature of route construction, and the notably success achieved by the classic AM [25] in vehicle routing problems (VRPs), we leverage a similar encoder-decoder structure as the policy network. However, unlike planning a route with the shortest distance in the vanilla AM, we aim to construct a route within the maximum flight distance of the UAV so as to execute as many tasks as possible. In this sense, we set the reward of the lower-level DRL model as the number of executed tasks for a UAV.

Similar to AM, the encoder-decoder architecture is used as the policy network for route construction. In the encoder, coordinates of allocated tasks and depot are first embedded with separate parameters, i.e., utilizing distinct linear projection layers for the initial embedding. After that, attention layers similar to the ones in the upper-level encoder are applied to further learn more informative embeddings for the tasks and depot. To avoid exceeding the maximum flight range of the UAV, we record and update the remaining flying distance during the route planning as follows,

$$D_{t+1} = D_t - d_{\pi_t,\pi_{t+1}}, \tag{19}$$

where D_t is the allowed remaining flight distance at time step t, $d_{\pi_t,\pi_{t+1}}$ is the distance between task π_t and π_{t+1}, and π_0 refers to the depot.

In the decoder, we construct the context at time step t with the graph embedding h_g, the embedding of the last task h_{t-1} and the remaining flight distance D_t. Then the *glimpse* attention [37] is applied to the decoder context and compatibilities are calculated. Finally, the probability distribution of executing the task at time step $t+1$ is derived through the softmax operation. Those produces could be briefly expressed as follows,

$$C_t^l = [h_g, h_{t-1}, D_t], \tag{20}$$

$$\widetilde{C}_t^l = glimpse\left(C_t^l\right), \tag{21}$$

$$u_t^l = C \cdot tanh(\frac{q_{lt}^T k_{lt}}{\sqrt{\dim_k}}), \tag{22}$$

$$p_t^l = softmax(u_t^l), \tag{23}$$

where u_t^l is the compatibilities calculated by a single-head attention operator; $q_{lt}^T = \widetilde{C}_t^l W_{com}^q$ and $k_{lt} = h^N W_{com}^k$ are the *query* and *key* vectors for the attention operation, with h^N being the embedding of the tasks, W_{com}^q and W_{com}^k being the trainable parameters; p_t^l is the probability vector of task selection at time step t.

For the convenience of batch training, some tasks might be repeatedly included in the task subset to a UAV, which helps keep the input length the same for the route construction DRL model. Therefore, to shrink the action space of the lower-level DRL model, we impose an initial mask matrix to prohibit the execution of these repetitive dummy tasks. Moreover, to ensure the feasibility of the final solution, during the UAV route construction, tasks that have been executed or cannot be executed within the remaining flight distance will also be masked at each time step.

C. Interactive Training Strategy

Within the DCF, we designed two different DRL models to solve the task allocation subproblem in the upper-level, and the UAV route planning subproblem in the lower-level, respectively. These two models may affect each other since the input of the lower-level model is determined by the upper-level model, and the reward of the upper-level model is calculated based on the output of the lower-level model. Therefore, we design an interactive training strategy (ITS) that consists of pre-training, intensive training, and alternate training processes to train them effectively and efficiently. The pseudocode of our ITS is presented in Algorithm 1. Pertaining to the training algorithm for our two models, we employ the rollout-based REINFORCE which performs well in training Transformer-style DRL models [40], [41], [48]. Accordingly, the gradient of the loss function in our two models is calculated as follows,

$$\nabla \mathcal{L}\left(\theta|s\right) = \mathbb{E}_{p_\theta(\pi|s)}[(L(\pi) - L(\pi^{BL}))\nabla_\theta \log p_\theta(\pi|s)], \tag{24}$$

where θ is the parameters of the policy network π, $L(\pi)$ and $L(\pi^{BL})$ are the cost function (i.e., the negative number of executed tasks) of the training model and the baseline model, respectively. We use the greedy decoding strategy in the baseline model. Its parameters will be replaced with that of the current training model if the current training model exhibits significant improvement according to the paired *t*-test. In this way, the loss variance will be potentially decreased during the whole training process.

Algorithm 1 Interactive training strategy

Input: the number of pre-training epochs E_p; the number of model training epochs E; the number of intensive training epochs E_t; the number of continuous training epochs E_c;
Initialization: the training datasets of the pre-training, intensive training and alternate training processes; the parameters of networks in the upper-level model M_u and the lower-level model M_l;

1: **for** $i = 1, 2, \ldots, E_p$ **do**
2: $L(\pi_l) \leftarrow -r_l(\mathrm{M}_l)$ ▷ r_l is the reward function
3: $L\left(\pi_l^{BL}\right) \leftarrow -r_l\left(\mathrm{M}_l^{BL}\right)$
4: $\mathrm{M}_l, \mathrm{M}_l^{BL} \leftarrow$ **REINFORCE**$(\mathrm{M}_l, L(\pi_l), L(\pi_l^{BL}))$
5: **end for**
6: **for** $epoch = 1, 2, \ldots, E$ **do**
7: $L(\pi_u) \leftarrow -r_u(\mathrm{M}_u, \mathrm{M}_l)$ ▷ r_u is the reward function
8: $L\left(\pi_u^{BL}\right) \leftarrow -r_u\left(\mathrm{M}_u^{BL}, \mathrm{M}_l\right)$
9: $\mathrm{M}_u, \mathrm{M}_u^{BL} \leftarrow$ **REINFORCE**$(\mathrm{M}_u, L(\pi_u), L(\pi_u^{BL}))$
10: **if** $epoch < E_t$ **then**
11: **if** $epoch \% E_c == 0$ **then**
12: $\mathrm{TD}_l \leftarrow data\_gen(\mathrm{M}_u)$
13: $\mathrm{TD}_l \leftarrow shuffle(\mathrm{TD}_l)$
14: $L(\pi_l) \leftarrow -r_l(\mathrm{M}_l)$
15: $L\left(\pi_l^{BL}\right) \leftarrow -r_l\left(\mathrm{M}_l^{BL}\right)$
16: $\mathrm{M}_l, \mathrm{M}_l^{BL} \leftarrow$ **REINFORCE**$(\mathrm{M}_l, L(\pi_l), L(\pi_l^{BL}))$
17: **end if**
18: **else**
19: $\mathrm{TD}_l \leftarrow data\_gen(\mathrm{M}_u)$
20: $\mathrm{TD}_l \leftarrow shuffle(\mathrm{TD}_l)$
21: $L(\pi_l) \leftarrow -r_l(\mathrm{M}_l)$
22: $L\left(\pi_l^{BL}\right) \leftarrow -r_l\left(\mathrm{M}_l^{BL}\right)$
23: $\mathrm{M}_l, \mathrm{M}_l^{BL} \leftarrow$ **REINFORCE**$(\mathrm{M}_l, L(\pi_l), L(\pi_l^{BL}))$
24: **end if**
25: **end for**

In our ITS, at the beginning of the models training, we construct the pre-training process for the lower-level model (lines 1-5), which is trained for E_p epochs. This process allows the lower-level model to learn an initial route planning policy to support the subsequent training process. After the pre-training process, both the upper-level and lower-level models are trained interactively, with the lower-level model fixed when training the upper-level model and vice versa. During the interactive training, there are two primary processes, 1) intensive training process (lines 7-17), in which the upper-level model is trained for E_c epochs continuously and the lower-level model is trained for the same number of epochs; 2) alternate training process (lines 7-9 and lines 19-23), in which the upper-level model and the lower-level model are trained for one epoch iteratively. In addition, due to the inherent correlation among the multiple instances for the lower-level model, which are generated from the same instance for the

upper-level model, we will randomly shuffle the generated instances for the lower-level model to mitigate the correlation.

V. Experiments and Analysis

In this section, we conduct extensive experiments to evaluate the performance of our DL-DRL for solving the multi-UAV task scheduling problem, and also compare it with various baselines. The effectiveness of the ITS is also assessed by comparing the convergence of the model with different training strategies. Besides, we further test our DL-DRL on instances of larger-scale to verify its generalization performance.

A. Experiment Settings

Following the convention in [25], [48], [49], we randomly generate the locations of tasks and depot in the square $[0, 1]$ following the uniform distribution. We consider two scenarios with 4 and 6 UAVs (termed U4 and U6), respectively, and each scenario includes small-scale instances, medium-scale instances, and large-scale instances according to the number of tasks. Instances with 80 and 100 tasks, 150 and 200 tasks, and 300 and 500 tasks are considered as small-scale, medium-scale, and large-scale, respectively. In this way, we could more comprehensively evaluate the overall performance of the approaches across different sizes of UAVs and tasks. Furthermore, the maximum flight distance of the UAV is set to 2.0 in all instances.

Regarding our DL-DRL, the number of attention layers in the encoder is set to $L = 3$. Both the upper-level and lower-level DRL models are trained for 100 epochs (except for pre-training) with 1,280,000 randomly generated instances per epoch. The batch size is set to 512 for the model training on small-scale and medium-scale instances, while 256 on the large-scale instances due to the memory constraint. We utilize the Adam optimizer to update the network parameters in the DRL models with an initial learning rate of 10^{-4}. In the upper-level model, the norm of grads is clipped within 3.0 and the decaying parameter of the learning rate is set to 0.995, while the same two hyperparameters are set to 1.0 in the lower-level model. In the ITS, the lower-level model is pre-trained for 5 epochs (i.e., $E_p = 5$), and different parameter settings are used for U4 and U6 scenarios during the interactive training. For the U4 scenario, the number of intensive training epochs E_t is set to 8, and the number of continuous training epochs E_c is set to 4, whereas E_t and E_c are set to 12 and 6 for the U6 scenario, respectively. In this way, the salient fluctuation of the model performance which results from the imbalanced training data of the upper-level and lower-level could be effectively mitigated throughout the training.

B. Learning Performance

Following the above experiment settings, our DL-DRL model is trained on U4 and U6 scenarios with different scales. For the training on the small-scale and medium-scale instances, a single RTX 3090 GPU with 24GB memory is used, whereas two GPUs are used for the training on the large-scale instances. The training time of the upper-level model, lower-level model, and data generation for the lower-level model for each epoch is displayed in Table I. In particular, "U4-80" means the model is trained on the U4 scenario with 80 tasks. Then we visualize the results of trained models on a number of exemplary instances in Fig. 3, where all the instances are randomly generated with seed 1234. Obviously, we can see that our trained DL-DRL models could deliver reasonably good solutions in various situations.

TABLE I: Time Consuming of Model Training Per Epoch (minutes)

Situation	Upper-level	Lower-level	Data generation
U4-80	20.55	6.12	13.02
U4-100	24.58	6.58	14.06
U4-150	32.15	8.78	19.30
U4-200	55.98	11.15	22.77
U4-300	123.70	23.32	45.50
U4-500	206.37	31.28	84.95
U6-80	25.92	6.03	17.19
U6-100	30.65	6.63	18.58
U6-150	49.25	8.55	23.35
U6-200	72.10	11.32	29.15
U6-300	155.55	22.92	58.49
U6-500	262.43	31.27	116.02

Meanwhile, we plot the learning curves of the upper-level and lower-level models during the training process in Fig. 4. The vertical axis refers to the cost value (i.e., negative of the total number of executed tasks), and the horizontal axis refers to the number of epoch. From Fig. 4, we can observe that, 1) For the lower-level model, it converges fast during the pre-training process, while the cost value dramatically increases at the beginning of the interactive training. This phenomenon might be caused by the difference in the training data distribution between pre-training and interactive training, i.e., the former is a uniform distribution, and the latter is unknown prior and produced by the upper-level model; 2) For the upper-level model, it exhibits several rapid dropping of cost value in the early training period. As the lower-level model performs better and better on the training data produced by the upper-level model during the intensive training process, the upper-level model also quickly delivers good performance. In addition, the learning curves demonstrate more obvious fluctuations on large-scale instances due to the expansion of the state and action space. Our DL-DRL models finally converge stably, suggesting that they have learned valid policies.

C. Comparison Analysis

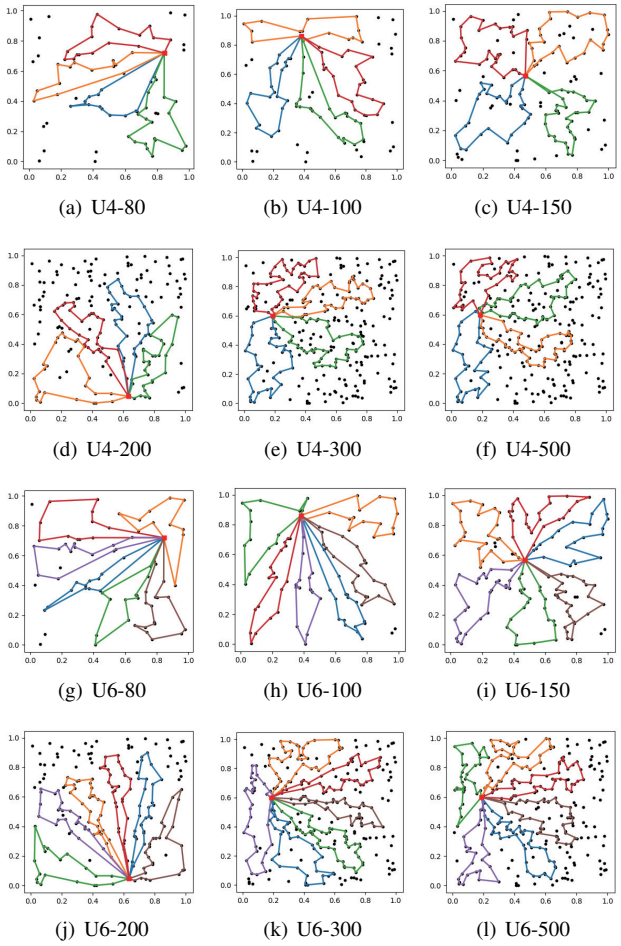
We compare the proposed DL-DRL approach with the exact solver, conventional heuristics, and learning-based baselines, which are briefly introduced as follows.

1) Gurobi[2]: A commercial exact solver for (mixed) integer programmings;

2) OR-Tools[3]: A widely used strong heuristic solver developed by Google;

[2]https://www.gurobi.com/

[3]https://developers.google.cn/optimization/



(a) U4-80 (b) U4-100 (c) U4-150

(d) U4-200 (e) U4-300 (f) U4-500

(g) U6-80 (h) U6-100 (i) U6-150

(j) U6-200 (k) U6-300 (l) U6-500

Fig. 3: Solution visualization of trained models in corresponding situations

3) K-means+VND: Based on the same DCF, it uses K-means and variable neighborhood descent (VND)[4] to handle task allocation and UAV route planning, respectively;

4) K-means+AM: Based on the same DCF, it uses K-means and AM [25] to handle task allocation and UAV route planning, respectively;

5) DL-DRL-I: An approach with the same network architecture as our DL-DRL, while it trains the two DRL models independently rather than using the proposed ITS.

As an exact solver, Gurobi is computationally time-consuming for solving multi-UAV task scheduling problems even on small-scale instances. Therefore, for each problem size, we evaluate Gurobi with 30 testing instances and limit the maximum solving time to 1800s, whereas other baselines are evaluated with 500 testing instances without any time limitation. All of the baselines are implemented in Python. The Intel Xeon Gold 5218R with 20 cores and RTX 3090 GPU are utilized for conventional and learning-based approaches, respectively. Regarding the upper-level model of DL-DRL and DL-DRL-I, both greedy and sampling strategies are employed in the decoder during the testing. According to Eq. (18), the greedy strategy always selects the UAV with the highest probability, whereas the sampling strategy engenders 128 solutions by sampling and reports the best. The experimental results on the U4 and U6 scenarios are presented in Table II and Table III, respectively. These tables gather the average objective value (Obj.), gap, and the average computation time (or runtime) of all methods. Here, the Obj. is the total number of completed tasks on average, and the gap is defined as the normalized distance between an average objective value Obj and the best one Obj_{best} found among all methods, which is expressed as follows,

$$Gap = \frac{Obj_{best} - Obj}{Obj_{best}} \times 100\%. \tag{25}$$

Regarding the testing on the 30 instances and 500 instances, the results are displayed in the left half and right half of the two tables, respectively. We can see from Tables II and III that Gurobi is unable to attain the optimal solution within the given time limit, even on instances with only 80 tasks. At the same time, our DL-DRL always achieves better solutions than Gurobi no matter which decoding strategy is used.

Pertaining to the U4 scenario in Table II, it is obvious that our DL-DRL (Greedy) outperforms K-means+VND and K-means+AM in terms of Obj. and computation time, with this superiority becoming more significant as the task scales up. Regarding the DL-DRL and DL-DRL-I, which have the same network architectures, the sampling strategy always yields better solutions than the greedy one, with only slightly longer computation time. Our DL-DRL (Greedy) outperforms DL-DRL-I (Greedy) and DL-DRL-I (Sampling), which implies the effectiveness of the ITS in improving the model performance. In comparison with the strong heuristic solver (i.e., OR-Tools), our DL-DRL (Sampling) is slightly inferior in terms of Obj. on the instances with the fewest tasks. However, as the task scales up, DL-DRL (Sampling) outperforms OR-Tools in terms of both Obj. and computation time, which justified the strong capability of our approach in tackling large-scale task scheduling problems of multi-UAV.

Pertaining to the U6 scenario in Table III, a similar pattern could be observed that our DL-DRL (Greedy and Sampling) outperforms K-means+VND, K-means+AM, DL-DRL-I (Greedy), and DL-DRL-I (Sampling). Meanwhile, our DL-DRL (Sampling) also exhibits a competitive performance against OR-Tools. On instances with no more than 150 tasks, DL-DRL (Sampling) is slightly inferior to OR-Tools in terms of Obj., but its computation time is much shorter. When the number of tasks increases, our DL-DRL (Sampling) outstrips OR-Tools in terms of both Obj. and computation time. Combining the results from both tables, DL-DRL demonstrates a better overall performance than Gurobi, K-means+VND, K-means+AM, and DL-DRL-I. It also outperforms OR-Tools on large-scale instances and performs competitively against OR-Tools with shorter computation time on small-scale instances.

D. Ablation Study on ITS

To investigate the efficacy of different processes in ITS, we construct the ablation study on the scenario U4 with 80 tasks. Different combinations of the training strategies are

[4]https://github.com/nchlpz/VND _TOP

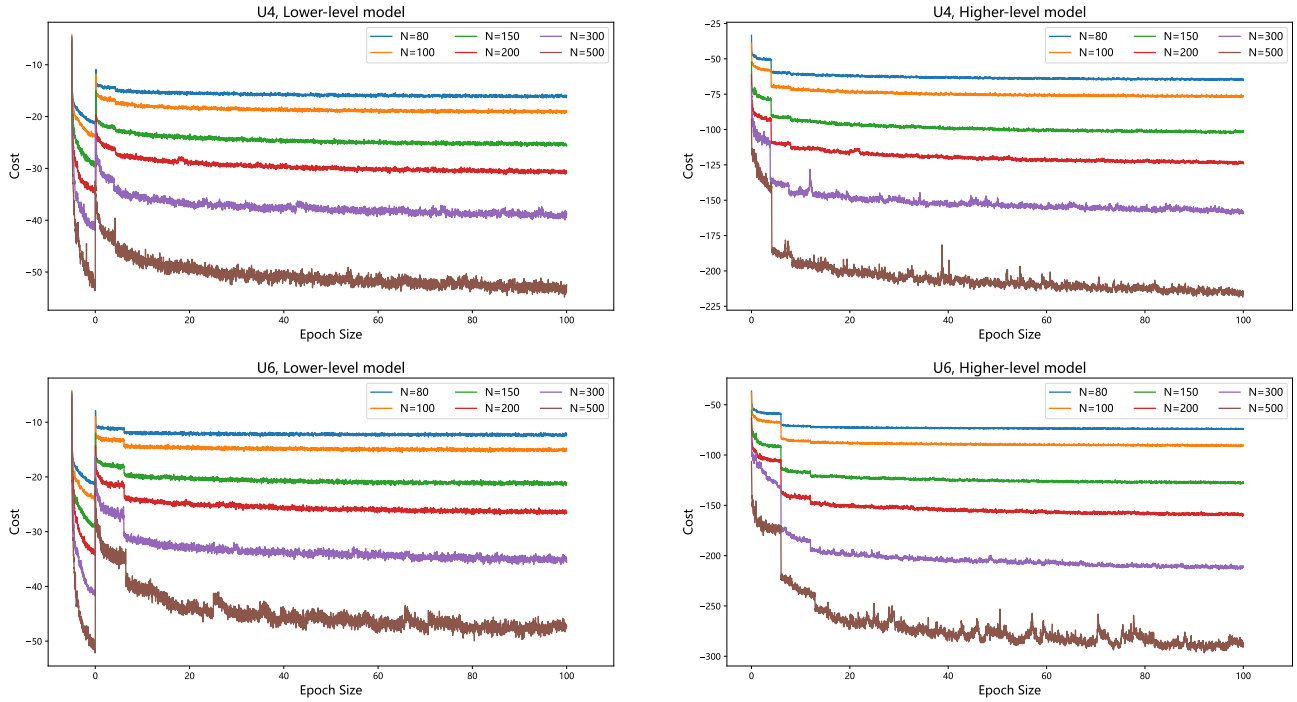


Fig. 4: Learning curves of DL-DRL

TABLE II: Experiment Results on the U4 Scenario

Number of tasks		Gurobi (1800s)	DL-DRL (Greedy)	DL-DRL (Sampling)	OR-tools	K-means +VND	K-means +AM	DL-DRL-I (Greedy)	DL-DRL-I (Sampling)	DL-DRL (Greedy)	DL-DRL (Sampling)
80	Obj.	56.77	63.67	**64.30**	**65.44**	56.45	59.68	58.78	61.32	64.67	65.28
	Gap	11.72	0.98	**0.00**	**0.00**	13.74	8.79	10.18	6.29	1.17	0.24
	Time(s)	1800.00	0.03	0.11	0.38	39.20	0.09	0.00	0.12	0.00	0.11
100	Obj.	65.37	76.20	**76.87**	76.83	62.65	69.35	67.61	71.14	76.67	**77.39**
	Gap	14.96	0.87	**0.00**	0.72	19.05	10.38	12.63	8.07	1.01	**0.00**
	Time(s)	1800.00	0.03	0.13	0.60	53.31	0.10	0.00	0.15	0.00	0.14
150	Obj.	75.10	102.57	**103.63**	101.34	74.11	87.14	90.28	96.41	102.25	**103.42**
	Gap	27.53	1.03	**0.00**	2.01	28.34	15.74	12.71	6.77	1.12	**0.00**
	Time(s)	1800.00	0.03	0.21	1.25	67.79	0.11	0.00	0.21	0.00	0.21
200	Obj.	74.37	74.37	**125.07**	121.21	81.47	100.85	106.68	115.34	123.76	**124.79**
	Gap	40.54	40.54	**0.00**	2.87	34.72	19.19	14.51	7.57	0.83	**0.00**
	Time(s)	1800.00	1800.00	0.29	2.00	96.51	0.12	0.00	0.30	0.00	0.27
300	Obj.	61.77	61.77	**159.23**	155.60	94.86	124.77	134.42	145.41	159.25	**160.14**
	Gap	61.21	61.21	**0.00**	2.84	40.77	22.09	16.06	9.20	0.56	**0.00**
	Time(s)	1800.00	1800.00	0.44	4.41	164.32	0.14	0.00	0.42	0.00	0.41
500	Obj.	69.33	69.33	**222.43**	209.00	113.88	154.71	170.20	188.78	216.66	**220.44**
	Gap	68.83	68.83	**0.00**	5.19	48.34	29.82	22.79	14.36	1.72	**0.00**
	Time(s)	1800.00	1800.00	0.89	11.71	316.45	0.15	0.01	0.84	0.01	0.85

used, and we record the cost value during the training process in Fig. 5. Particularly, "ITS/X" refers to our ITS but without the "X" component. For example, "ITS/intensive" denotes the training strategy that only includes the pre-training and alternate training processes.

From Fig. 5, we can observe that the pre-trained models perform better at the beginning of the interactive training (e.g., "ITS" versus "ITS/pre-training"). The intensive training process slows down the convergence by several epochs (e.g., "ITS/pre-training" versus "ITS/pre-training&intensive"). However, as indicated in Table I, for each epoch, the time used to generate the training data for the lower-level model is much longer than that of the lower-level model training (e.g., 116 minutes versus 31 minutes for U6-500). The intensive training process does not need this time-consuming training data generation for the lower-level model, which considerably reduces the whole training time. With respect to the eventual performance, our ITS exhibits a similar result to ITS/intensive, whereas the whole training time of ITS (ours) is much shorter than that of ITS/intensive due to the less lower-level training data generation. Thus, we use the proposed ITS which comes with an intensive training process to accelerate the entire training process without performance drop.

TABLE III: Experiment Results on the U6 Scenario

Number of tasks		Gurobi (1800s)	DL-DRL (Greedy)	DL-DRL (Sampling)	OR-tools	K-means +VND	K-means +AM	DL-DRL-I (Greedy)	DL-DRL-I (Sampling)	DL-DRL (Greedy)	DL-DRL (Sampling)
80	Obj.	64.37	73.10	**73.53**	**75.42**	68.59	68.31	68.91	71.01	74.00	74.32
	Gap	12.47	0.59	**0.00**	**0.00**	9.05	9.43	8.63	5.85	1.88	1.46
	Time(s)	1800.00	0.03	0.15	0.50	20.65	0.14	0.00	0.13	0.00	0.14
100	Obj.	73.23	90.27	**91.23**	**92.42**	81.38	82.12	82.60	85.43	90.72	91.53
	Gap	19.73	1.06	**0.00**	**0.00**	11.95	11.14	10.63	7.56	1.84	0.96
	Time(s)	1800.00	0.03	0.19	0.85	36.32	0.16	0.00	0.19	0.00	0.17
150	Obj.	79.63	128.23	**129.60**	**130.09**	104.49	110.14	110.53	116.29	128.18	129.59
	Gap	38.56	1.06	**0.00**	**0.00**	19.67	15.33	15.04	10.60	1.46	0.38
	Time(s)	1800.00	0.03	0.27	2.21	88.55	0.19	0.00	0.26	0.01	0.24
200	Obj.	73.27	157.37	**159.97**	160.79	117.54	136.05	137.09	144.78	158.96	**160.83**
	Gap	54.20	1.63	**0.00**	0.03	26.92	15.41	14.76	9.98	1.16	**0.00**
	Time(s)	1800.00	0.04	0.37	3.73	151.36	0.23	0.00	0.35	0.01	0.35
300	Obj.	79.80	211.93	**214.27**	212.46	135.89	165.10	178.61	191.23	213.27	**215.47**
	Gap	62.76	1.09	**0.00**	1.40	36.94	23.38	17.11	11.25	1.02	**0.00**
	Time(s)	1800.00	0.05	0.58	8.00	221.86	0.28	0.01	0.57	0.01	0.53
500	Obj.	61.00	295.40	**299.73**	292.10	162.25	193.21	201.76	226.21	293.18	**297.59**
	Gap	79.65	1.45	**0.00**	1.84	45.48	35.07	32.20	23.99	1.48	**0.00**
	Time(s)	1800.00	0.06	1.18	20.84	382.92	0.32	0.01	1.07	0.01	1.05

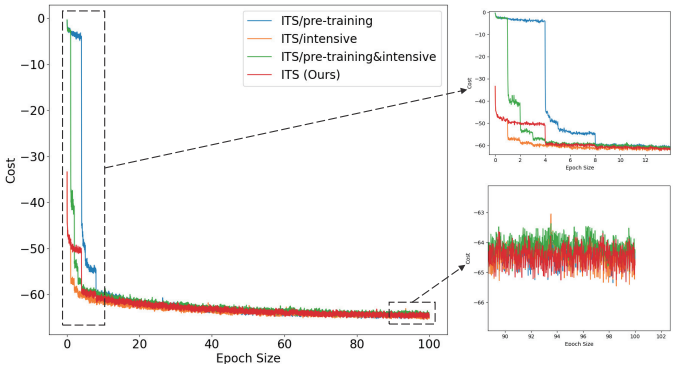


Fig. 5: Ablation study results of ITS

E. Generalization to Larger-scale Instances

To verify the generalization performance of our DL-DRL, we use the trained models to solve larger-scale instances. The DL-DRL (Sampling) is adopted since the sampling strategy can obtain better solutions with only slightly longer time than the greedy one. We first generalize the trained DL-DRL models from subsection V-B to solve the problem with a larger task scale. The results are shown in Fig. 6 and 7, respectively, where the horizontal coordinate refers to the problems that need to be solved, and the legend denotes the trained models. For example, N-100 represents the problem with 100 tasks, and U4-80 represents the DL-DRL model trained on U4 with 80 tasks.

From Fig. 6, we can see that the model trained for a certain problem size performs best on the corresponding problem compared to those trained for other problem sizes. However, they still outperform K-means+VND, K-means+AM, and DL-DRL-I except for U4-80 in solving problems with no less than 200 tasks and U4-100 in solving the problem with 500 tasks. In addition, we notice that the model trained for proximal problem sizes performs better than that of different ones (e.g., U4-200 and U4-300 perform better than U4-80, U4-100, and U4-150 in solving problems with 500 tasks). This phenomenon might be caused by the difference in data distribution, as the proximal problem sizes may lead to similar data distributions of task location. A similar pattern can also be found in Fig. 7, where the models trained for other problem sizes outperform K-means+VND and K-means+AM and are competitive against DL-DRL-I.

For larger-scale multi-UAV task scheduling problems with like 1000 tasks, it is intractable to train the model from scratch, rendering the generalizability a highly desired capability for the trained models. To further investigate the generalization performance of the proposed DL-DRL, we apply the one trained with 500 tasks to solve the instances with 600, 700, 800, 900, and 1000 tasks, and also compare it with the baselines including OR-Tools, K-means+AM, and DL-DRL-I. Table IV displays the testing results of our DL-DRL and the baselines, including Obj. and average computation time. Compared to K-means+AM and DL-DRL-I, a salient superiority of DL-DRL can be observed in terms of Obj. with slightly longer computation time. Regarding OR-Tools, its computation time increases rapidly as the task scales up, and our DL-DRL exceeds it in terms of both Obj. and computation time. On all the scenarios, our DL-DRL achieves the best overall performance among all studied methods, which justifies its favorable generalization capability.

VI. Conclusion

In this paper, we investigate the DRL approach for the large-scale multi-UAV task scheduling problem. Based on the divide and conquer framework (DCF), the original problem is decomposed into task allocation subproblem and UAV route planning subproblem, and we proposed a double-level DRL (DL-DRL) approach to solve them. In our DL-DRL, two different DRL

TABLE IV: Generalization results of DL-DRL and baselines

	Method	N=600		N=700		N=800		N=900		N=1000	
		Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)
U4	OR-tools	231.79	16.93	251.65	22.15	271.20	28.50	288.36	36.17	305.64	43.01
	K-means+AM	165.73	**0.27**	174.75	**0.28**	181.54	**0.29**	187.72	**0.30**	192.43	**0.31**
	DL-DRL-I	206.46	1.06	220.48	1.31	232.94	1.61	241.13	1.93	248.81	2.28
	DL-DRL	**245.67**	1.07	**267.88**	1.33	**287.36**	1.62	**303.83**	1.95	**321.58**	2.34
U6	OR-tools	326.37	30.45	356.94	39.70	386.45	52.30	411.60	65.73	437.17	79.83
	K-means+AM	209.12	**0.35**	221.79	**0.37**	230.78	**0.38**	238.52	0.39	244.79	**0.39**
	DL-DRL-I	254.55	1.47	277.81	1.78	295.52	2.21	305.69	2.67	316.94	3.24
	DL-DRL	**334.28**	1.43	**367.61**	1.84	**394.90**	2.28	**419.26**	2.74	**443.20**	3.32

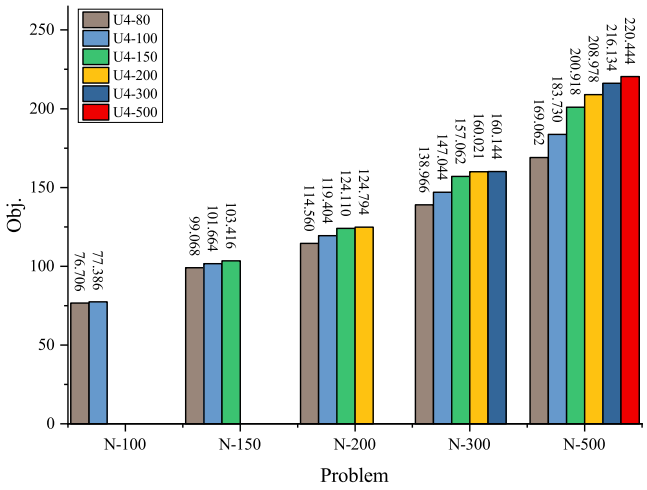


Fig. 6: Generalization to larger instances on U4

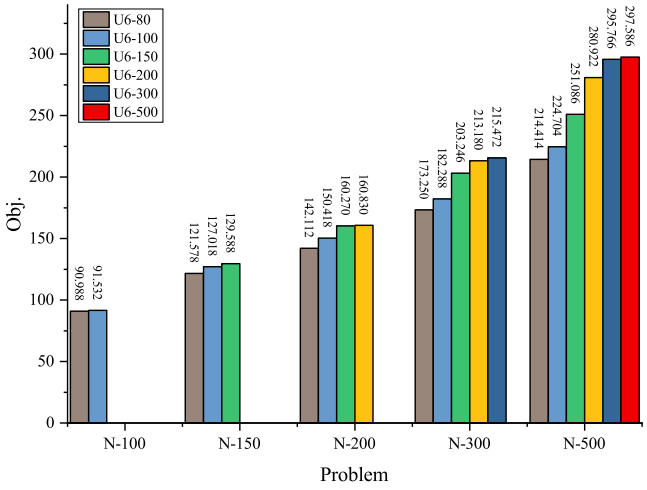


Fig. 7: Generalization to larger instances on U6

models are exploited to solve the two subproblems in the upper level and the lower level, respectively. Considering the intrinsic relationship of subproblems, an interactive training strategy (ITS) is designed for effective training, which comprises pre-training, intensive training, and alternate training. The experimental results show that our DL-DRL achieves the best overall performance compared to exact solver, conventional heuristic, and learning-based baselines, especially on large-scale instances. Furthermore, the generalization of our DL-DRL to larger-scale problems and the effectiveness of our ITS are also verified through experiments, respectively. Regarding the future studies, we will further improve our approach to tackle various distributions of task locations, test on real-world instances, and compare with other strong heuristic methods.

VII. Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62073341 and the Fundadmental Reasearch Funds for the Central Universities of Central South University (No. 2022ZZTS0659).

References

[1] Y. Liu, "An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones," *Computers & Operations Research*, vol. 111, pp. 1–20, Nov. 2019.

[2] D. Machovec, J. A. Crowder, H. J. Siegel, S. Pasricha, and A. A. Maciejewski, "Dynamic Heuristics for Surveillance Mission Scheduling with Unmanned Aerial Vehicles in Heterogeneous Environments," in *Advances in Artificial Intelligence and Applied Cognitive Computing*. Cham: Springer International Publishing, 2021, pp. 583–605.

[3] A. Altan and R. Hacıoğlu, "Model predictive control of three-axis gimbal system mounted on UAV for real-time target tracking under external disturbances," *Mechanical Systems and Signal Processing*, vol. 138, p. 106548, Apr. 2020.

[4] Z. Wang, L. Liu, T. Long, and Y. Wen, "Multi-UAV reconnaissance task allocation for heterogeneous targets using an opposition-based genetic algorithm with double-chromosome encoding," *Chinese Journal of Aeronautics*, vol. 31, no. 2, pp. 339–350, Feb. 2018.

[5] G. Laporte and Y. Nobert, "A Cutting Planes Algorithm for the m-Salesmen Problem," *Journal of the Operational Research Society*, vol. 31, no. 11, pp. 1017–1023, Nov. 1980.

[6] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, Sep. 2004.

[7] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and Control of Multiple UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2002.

[8] M. Alighanbari and J. How, "Cooperative task assignment of unmanned aerial vehicles in adversarial environments," in *Proceedings of the 2005, American Control Conference, 2005.* Portland, OR, USA: IEEE, 2005, pp. 4661–4666.

[9] E. J. Forsmo, E. I. Grøtli, T. I. Fossen, and T. A. Johansen, "Optimal search mission with Unmanned Aerial Vehicles using Mixed Integer Linear Programming," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2013, pp. 253–259.

[10] E. Edison and T. Shima, "Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms," *Computers & Operations Research*, vol. 38, no. 1, pp. 340–356, Jan. 2011.

[11] S. Gao, J. Wu, and J. Ai, "Multi-UAV reconnaissance task allocation for heterogeneous targets using grouping ant colony optimization algorithm," *Soft Computing*, vol. 25, no. 10, pp. 7155–7167, May 2021.

[12] N. Geng, Z. Chen, Q. A. Nguyen, and D. Gong, "Particle swarm optimization algorithm for the optimization of rescue task allocation with uncertain time constraints," *Complex & Intelligent Systems*, vol. 7, no. 2, pp. 873–890, Apr. 2021.

[13] L. Huo, J. Zhu, G. Wu, and Z. Li, "A Novel Simulated Annealing Based Strategy for Balanced UAV Task Assignment and Path Planning," *Sensors*, vol. 20, no. 17, p. 4769, Jan. 2020.

[14] H. Liu, X. Li, G. Wu, M. Fan, R. Wang, L. Gao, and W. Pedrycz, "An iterative two-phase optimization method based on divide and conquer framework for integrated scheduling of multiple UAVs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5926–5938, Sep. 2021.

[15] F. Zitouni, S. Harous, and R. Maamri, "A Distributed Solution to the Multi-robot Task Allocation Problem Using Ant Colony Optimization and Bat Algorithm," in *Advances in Machine Learning and Computational Intelligence*. Singapore: Springer, 2021, pp. 477–490.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] Y. Zhu and D. Zhao, "Online Minimax Q Network Learning for Two-Player Zero-Sum Markov Games," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 3, pp. 1228–1241, Mar. 2022.

[18] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 737–744.

[19] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, "A Survey of Reinforcement Learning Informed by Natural Language," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 6309–6317.

[20] H. Lu, X. Zhang, and S. Yang, "A Learning-based Iterative Method for Solving Vehicle Routing Problems," in *International Conference on Learning Representations*, Sep. 2019.

[21] J. Zheng, K. He, J. Zhou, Y. Jin, and C.-M. Li, "Combining Reinforcement Learning with Lin-Kernighan-Helsgaun Algorithm for the Traveling Salesman Problem," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, pp. 12 445–12 452, May 2021.

[22] R. Gama and H. L. Fernandes, "A reinforcement learning approach to the orienteering problem with time windows," *Computers & Operations Research*, vol. 133, p. 105357, Sep. 2021.

[23] Y. Ma, X. Hao, J. Hao, J. Lu, X. Liu, T. Xialiang, M. Yuan, Z. Li, J. Tang, and Z. Meng, "A Hierarchical Reinforcement Learning Based Optimization Framework for Large-scale Dynamic Pickup and Delivery Problems," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 23 609–23 620.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010.

[25] W. Kool, H. van Hoof, and M. Welling, "ATTENTION, LEARN TO SOLVE ROUTING PROBLEMS!" in *International Conference on Learning Representations*, 2019.

[26] P.-L. Bacon, J. Harb, and D. Precup, "The Option-Critic Architecture," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017.

[27] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-Efficient Hierarchical Reinforcement Learning," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[28] C. Ramirez-Atencia, G. Bello-Orgaz, M. D. R-Moreno, and D. Camacho, "Performance Evaluation of Multi-UAV Cooperative Mission Planning Models," in *Computational Collective Intelligence*. Cham: Springer International Publishing, 2015, pp. 203–212.

[29] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1093–1099.

[30] F. Mufalli, R. Batta, and R. Nagi, "Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans," *Computers & Operations Research*, vol. 39, no. 11, pp. 2787–2799, Nov. 2012.

[31] F. Ye, J. Chen, Y. Tian, and T. Jiang, "Cooperative Task Assignment of a Heterogeneous Multi-UAV System Using an Adaptive Genetic Algorithm," *Electronics*, vol. 9, no. 4, p. 687, Apr. 2020.

[32] K. Shang, S. Karungaru, Z. Feng, L. Ke, and K. Terada, "A GA-ACO hybrid algorithm for the multi-UAV mission planning problem," in *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, Sep. 2014, pp. 243–248.

[33] Y. Chen, D. Yang, and J. Yu, "Multi-UAV Task Assignment With Parameter and Time-Sensitive Uncertainties Using Modified Two-Part Wolf Pack Search Algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 6, pp. 2853–2872, Dec. 2018.

[34] S. Gupta, G. Singal, and D. Garg, "Deep Reinforcement Learning Techniques in Diversified Domains: A Survey," *Archives of Computational Methods in Engineering*, vol. 28, no. 7, pp. 4715–4754, Dec. 2021.

[35] S. Luo, L. Zhang, and Y. Fan, "Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3020–3038, 2021.

[36] Y. Wang, S. Sun, and W. Li, "Hierarchical Reinforcement Learning for Vehicle Routing Problems with Time Windows," *Proceedings of the Canadian Conference on Artificial Intelligence*, Jun. 2021.

[37] I. Bello, H. Pham, Q. Le, M. Norouzi, and S. Bengio, "Neural Combinatorial Optimization with Reinforcement Learning," Feb. 2017.

[38] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer Networks," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015.

[39] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial Optimization by Graph Pointer Networks and Hierarchical Reinforcement Learning," *arXiv:1911.04936 [cs, stat]*, Nov. 2019.

[40] Y. Xu, M. Fang, L. Chen, G. Xu, Y. Du, and C. Zhang, "Reinforcement Learning With Multiple Relational Attention for Solving Vehicle Routing Problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 11 107–11 120, Oct. 2022.

[41] Y. Hu, Y. Yao, and W. S. Lee, "A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs," *Knowledge-Based Systems*, vol. 204, p. 106244, Sep. 2020.

[42] W. Liu, T. Zhang, R. Wang, K. Li, W. Li, and K. Yang, "Deep Reinforcement Learning for Orienteering Problems Based on Decomposition," Apr. 2022.

[43] X. Xu, H. Yuan, M. Liptrott, and M. Trovati, "Two phase heuristic algorithm for the multiple-travelling salesman problem," *Soft Computing*, vol. 22, no. 19, pp. 6567–6581, Oct. 2018.

[44] I. Dolinskaya, Z. E. Shi, and K. Smilowitz, "Adaptive orienteering problem with stochastic travel times," *Transportation Research Part E: Logistics and Transportation Review*, vol. 109, pp. 1–19, Jan. 2018.

[45] I. V. Tetko, P. Karpov, R. Van Deursen, and G. Godin, "State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis," *Nature Communications*, vol. 11, no. 1, p. 5575, Nov. 2020.

[46] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, Sep. 2020.

[47] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1441–1450.

[48] J. Li, Y. Ma, R. Gao, Z. Cao, A. Lim, W. Song, and J. Zhang, "Deep Reinforcement Learning for Solving the Heterogeneous Capacitated Vehicle Routing Problem," *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

[49] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takac, "Reinforcement Learning for Solving the Vehicle Routing Problem," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

Xiao Mao received the B.S. degree in transportation engineering from Central South University, Changsha, China, in 2020. He is currently pursuing the Ph.D. degree in School of Traffic and Transportation Engineering, Central South University. His research interests include reinforcement learning and intelligent scheduling.

Zhiguang Cao received the Ph.D. degree from Interdisciplinary Graduate School, Nanyang Technological University. He received the B.Eng. degree in Automation from Guangdong University of Technology, Guangzhou, China, and the M.Sc. in Signal Processing from Nanyang Technological University, Singapore, respectively. He was a Research Fellow with the Energy Research Institute @ NTU (ERI@N), a Research Assistant Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, and a Scientist with the Agency for Science Technology and Research (A*STAR), Singapore. He joins the School of Computing and Information Systems, Singapore Management University, as an Assistant Professor. His research interests focus on learning to optimize (L2Opt).

Mingfeng Fan received the B.S. degree in transport equipment and control engineering from Central South University, Changsha, China, in 2019, where she is currently pursuing the Ph.D. degree in traffic and transportation engineering. Her research interests include machine learning and UAV path planning.

Guohua Wu (Member, IEEE) received the B.S. degree in Information Systems and Ph.D degree in Operations Research from National University of Defense Technology, China, in 2008 and 2014, respectively. During 2012 and 2014, he was a visiting Ph.D student at University of Alberta, Edmonton, Canada. He is now a Professor at the School of Traffic and Transportation Engineering, Central South University, Changsha, China.

His current research interests include scheduling, computational intelligence, and machine learning. He has authored more than 100 referred papers including those published in *IEEE TCYB*, *IEEE TEVC* and *IEEE TSMCA*. He serves as an Associate Editor of *Information Sciences*, and *Swarm and Evolutionary Computation*, an editorial board member of *International Journal of Bio-Inspired Computation*, a Guest Editor of *Information Sciences and Memetic Computing*.

Witold Pedrycz (Life Fellow, IEEE) is a Professor and the Canada Research Chair (CRC-Computational Intelligence) with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, and also with the Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah, Saudi Arabia. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. In 2012 he was elected a Fellow of the Royal Society of Canada.

His current research interests include computational intelligence, knowledge discovery and data mining, pattern recognition, knowledge-based neural networks, and software engineering. He has published numerous papers in the above areas. He is an Editor-in-Chief of *Information Sciences*. He currently serves as an Associate Editor of *IEEE Transactions on Fuzzy Systems* and *IEEE Transactions on Systems, Man and Cybernetics: System* and is a member of a number of editorial boards of other international journals.