DL-DRL: A Double-Level Deep Reinforcement Learning Approach for Large-Scale Task Scheduling of Multi-UAV

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

Abstract—Exploiting unmanned aerial vehicles (UAVs) to execute tasks is gaining growing popularity recently. To address the underlying task scheduling problem, conventional exact and heuristic algorithms encounter challenges such as rapidly increasing computation time and heavy reliance on domain knowledge, particularly when dealing with large-scale problems. The deep reinforcement learning (DRL) based methods that learn useful patterns from massive data demonstrate notable advantages. 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 total value 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 1500 tasks and to different flight distances of UAVs. 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 at https://faculty.csu.edu.cn/guohuawu/zh_CN/zdylm/193832/list/index.htm.

Note to Practitioners—Unmanned aerial vehicles (UAVs) are of great practical usage, as they have many real world applications. When a group of UAVs are employed to execute large-scale tasks, a core question is how to scheduling the UAVs, so that they could complete the tasks efficiently. However, it is a computationally hard problem due to the exponentially increasing search space. To solve this problem, we propose a double-level deep reinforcement learning (DL-DRL) approach within a divide-and-conquer framework (DCF), where the upper-level DRL model is responsible for the task allocation, and the lower-level DRL model is responsible for the UAV route planning. To better train the two DRL models who have interplay with each other, we propose a simple yet efficient training strategy, termed interactive training strategy (ITS), which includes pre-training, intensive training and alternate training. The experimental results based on instances of various scales show that our DL-DRL approach outperformed learning-based and conventional baselines, and the designed ITS could strike a good balance between performance and training efficiency. In light of those verified advantages, we believe that our DL-DRL approach has favorable potential to solve the practical task scheduling problem of multi-UAV in real world.

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. In this paper, considering the rapidly increasing execution task quantity for multi-UAV systems, we study the large-scale task scheduling problem of multi-UAV, where tasks with the attributes: 1) task value, which reflects the importance of the task, it can be the same or different among all the tasks according to the reality [5], [6], [7]; 2) position, which is...
expressed by the coordinates. All the tasks are independent and appear before UAVs take off, and UAVs have the maximum flight distance due to the energy capacity.

As a variant of the multiple traveling salesman problem (m-TSP) [8], the multi-UAV task scheduling problem is known to be NP-hard [9], for which it is difficult to find an optimal solution in a reasonable computation time, especially in large-scale situations. Both exact and heuristic algorithms have been investigated and explored to solve this problem, whereas exact algorithms [10], [11], [12] can deliver optimal solutions on relatively small-scale problems and are incapable on large-scale ones due to the exponentially increasing computation time. Heuristic algorithms (e.g., genetic algorithm (GA) [13], ant colony optimization (ACO) [14], particle swarm optimization (PSO) [15], and simulated annealing algorithm (SA) [16]) are desirable to tackle the large-scale problem. However, it is hard to strike a balance between solution quality and computation efficiency. In addition, 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 of multi-UAV, a number of recent studies decompose the original problem into several subproblems, which are then solved by conventional heuristics [17], [18]. In this way, the computation complexity could be effectively reduced, thus making the decomposition scheme more favorable for handling large-scale problems. However, the handcrafted 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 in deep learning (DL) and reinforcement learning (RL), deep reinforcement learning (DRL) has been widely explored in games [19], [20], robotics [21], and natural language processing [22]. Recently, DRL has also been popularly applied to combinatorial optimization problems (COPs), such as capacitated vehicle routing problem (CVRP) [23], traveling salesman problem (TSP) [24], and orienteering problem (OP) [25].

DRL approaches can automatically learn a useful decision-making policy in a data-driven manner by leveraging the underlying pattern among the COP instances. In this way, they can generalize to unseen instances and generate high-quality solutions in a short time due to the efficient inference process. Considering the large-scale task scheduling scenarios of multi-UAV, it is common to face high-dimensional state and action space caused by multiple tasks, UAVs, and their associated attributes, such as locations. In contrast to heuristic algorithms, which insert problem-specific handcrafted rules to simplify the complex solution space, DRL approaches can effectively handle such large-scale inputs and learn the corresponding policy to generate task scheduling schemes for multi-UAVs. Considering the above advantages, we exploit DRL to solve the large-scale task scheduling problem of multi-UAV. However, the decision space in DRL expands sharply with the scale of the problem grows, which may lead to unstable training and undesirable performance [26].

In this paper, we exploit a divide and conquer framework (DCF) to decompose the original task scheduling problem of multi-UAV into a task allocation subproblem and a UAV route planning subproblem. As a result, the decision space of subproblems is reduced compared to the original problem, which effectively alleviates the training instability of DRL models in the huge decision space. Then, given the decomposed decision space, we propose a double-level deep reinforcement learning approach (DL-DRL) to solve the two subproblems, respectively, in which DRL models can automatically learn useful decision-making policies instead of making handcrafted rules in heuristic algorithms. In particular, the upper-level DRL model is mainly responsible for the task allocation, in which an encoder based on the self-attention mechanism [27] 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) [28] is designed. Unlike the general hierarchical DRL, which needs to learn the goal/option determination by trial and error [29], [30], the goals of our DL-DRL are determined by the DCF in advance. In this way, policy learning is reduced to only action selection at both levels, making the training of DRL models more efficient and stable. 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, as well as its desirable generalization to larger problem instances and different flight distances of UAVs. 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 learning-based and conventional baselines. The efficacy of our two DRL models (i.e., upper-level and lower-level models) and training strategy are also justified through ablation studies. Additionally, by applying the model learned for a problem size to solve larger ones (up to 1500 tasks) and for different flight distances of UAVs, our
DRL approach 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. Section IV presents the divide and conquer framework for problem decomposition and 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 [10], [31] for this problem and design exact or heuristic algorithms to solve it. In this section, we review related works from the aspect of solving methods. Introducing and analyzing the advantages and shortages of relevant traditional exact and heuristic algorithms first, and then reviewing DRL for solving COPs. Based on the review, we conclude the characteristics of the existing methods for solving multi-UAV task scheduling problems and further introduce the decomposition methods for large-scale task scheduling problems.

Exact algorithms such as branch and bound [32], branch and price [33], column generation [34], and dynamic programming [11] have the potential to attain optimal solutions, whereas they are unsuitable to solve large-scale problems due to the prohibitive computation time for exhaustive search. Heuristic algorithms that can reduce search space with carefully designed heuristic rules are reckoned as alternatives for solving the multi-UAV task scheduling problem. Ye et al. [35] developed an GA method with a multi-type gene chromosome encoding scheme and an adaptive operation for efficient solution searching. Shang et al. [36] 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. [37] proposed a modified two-part wolf pack search (MTWPS) algorithm based on a designed transposition and extension operation to solve the problem. However, the handcrafted rules highly rely on human experience and domain knowledge, the quality of solutions generated by heuristic algorithms 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 [38] due to its powerful learning capability. As DRL can automatically learn a useful decision-making policy via the interaction with the environment, it has been recently applied to solve COPs [39], [40], which learns the rules from a large number of problem instances rather than manually designing them. Bello et al. [41] presented a DRL model based on pointer networks (PtrNet) [42] 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. [43] proposed a graph pointer network that integrates the graph neural network and PtrNet for better feature extraction. In addition, Kool et al. [28] proposed the AM (Attention Model) method based on the Transformer architecture [27] to solve a variety of COPs, which outperforms a wide range of learning-based and conventional baselines. Xu et al. [44] improved the AM to solve VRPs more efficiently by incorporating a multiple relational attention mechanism. Zhao et al. [45] constructed a DRL model combining AM with a pointer mechanism to solve the flexible job shop scheduling problem. Luo et al. [46] constructed a light encoder and heavy decoder model, which modified the AM by reducing and adding attention layers in the encoder and decoder, respectively. These DRL methods show 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 [26].

Among existing approaches for task scheduling problems of multi-UAV, traditional exact and heuristic algorithms are frustrating in addressing large-scale problems. Exact algorithms are unbearably computationally expensive when faced with large-scale problems, while heuristic algorithms struggle to guarantee the solution quality due to their heavy reliance on handcrafted rules, and iterative search is also time-consuming for large-scale problems. Hence, when the task scale increases, both exact and heuristic algorithms would experience a significant decline in performance. 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 the above issue, recent works have concentrated on the problem decomposition. Hu et al. [47] divided the m-TSP into several small-scale TSP via the proposed DRL policy networks. Hou et al. [48] proposed a two-stage divide method to decompose the VRP into a customer dividing subproblem and a TSP subproblem, leveraging a two-step RL for training the customer dividing model and the TSP solving model. Moreover, Chen et al. [49] deemed each UAV as an agent and proposed a global-and-local attention-based DRL approach for the task execution of multi-UAV, where global and local attention mechanisms are used for the whole multi-UAV system and single UAV, respectively. In this way, large-scale problems are 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. Based on DCF, a DL-DRL approach is proposed to solve the two subproblems at the upper and lower levels, respectively. Furthermore, an efficient training strategy, termed ITS, is designed for the effective training of the double-level DRL models. Through the decomposition of the problem, the DRL model construction for subproblems, and the efficient model training, our DL-DRL has favorable potential for solving large-scale multi-UAV task scheduling problems.
TABLE I
DEFINITIONS OF NOTATIONS

| Notations | Description |
|-----------|-------------|
| $U$       | UAV set, $U = \{1, 2, \cdots, V\}$ |
| $k$       | UAV index |
| $T$       | Task set, $T = \{1, 2, \cdots, N\}$ |
| $i, j$    | Task index |
| $x_{i,j}^k$ | A binary variable that equals 1 if the UAV $k \in U$ executes the task $j$ right after the task $i$ and 0 otherwise, $(i, j) \in T$. |
| $y_i^k$   | A binary variable that equals 1 if the UAV $k \in U$ has executed the task $i$ in $T$ and 0 otherwise. |
| $z_{i,j}^k$ | A continuous variable indicating the allowed remaining flight distance of the UAV $k \in U$ when visiting task $i \in T$. |
| $X_i, Y_i$ | The coordinates of the task $i$. |
| $D$       | Maximum flight distance of UAV |
| $d_{i,j}$ | Distance between task $i$ and $j$. |
| $r_i$     | The value of task $i$. |

III. PROBLEM FORMULATION

Consider a fleet of $V$ homogeneous UAVs with the same maximum flight distance initially located in the UAV depot. They need to execute given tasks while trying to maximize the total value of tasks. We assume that $N$ independent tasks are distributed in an area, and all UAVs maintain a constant speed while flying at the same and fixed altitude. Offline and non-preemptive scheduling are mainly discussed in our study. All notations used for formulating the task scheduling problem are listed in Table I.

A. Task Model

For each task, it can be executed once at most among all UAVs, and the following constraint should be satisfied:

$$C1: \sum_{k \in U} y_i^k \leq 1, \quad i \in T$$  

(1)

The distance between any two tasks is expressed as:

$$d_{i,j} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}, \quad i, j \in T$$  

(2)

B. UAV Model

Let index 0 and $N + 1$ denote the UAV depot, which each UAV must depart from and finally return to after executing the corresponding tasks. That is, the following constraint should be satisfied:

$$C2: \sum_{k \in U} \sum_{j \in T} x_{0,j}^k = \sum_{k \in U} \sum_{i \in T} x_{i,N+1}^k = V$$  

(3)

When the UAV needs to execute the task $j$, it will depart from the current position (i.e., the location of the last task) and will also depart from the task $j$ to execute the next task. Thus, another constraint holds:

$$C3: \sum_{j \in T \cup \{N+1\}} x_{i,j}^k = \sum_{j \in T \cup \{0\}} x_{i,j}^k \leq 1, \quad i \in T \cup \{0, N+1\}, \quad k \in U$$  

(4)

Note that the route length of the UAV cannot exceed its maximum flight distance. That is:

$$C4: \sum_{i \in T \cup \{0, N+1\}} \sum_{j \in T \cup \{0, N+1\}} d_{i,j} \cdot x_{i,j}^k \leq D, \quad k \in U$$  

(5)

In addition, the UAV cannot conduct flights with subtours, i.e., depart from one task to execute other tasks and subsequently return to the initial task. This constraint is given as [50]:

$$C5: z_{i,j}^k - z_{i,j}^k + D \cdot x_{i,j}^k \leq D - d_{i,j}, \quad i \neq j \in T \cup \{0, N+1\}, \quad k \in U$$  

(6)

C. Mathematical Model

The objective of the task scheduling problem for multi-UAV is to determine the order of task execution for each UAV in a way to achieve the maximum total value of executed tasks. Thus, the overall mathematical model can be formulated as follows:

$$\text{max} \sum_{i \in T} \sum_{k \in U} r_i \cdot y_i^k$$  

s.t. $C1: \sum_{k \in U} y_i^k \leq 1, \quad i \in T$

$$C2: \sum_{k \in U} \sum_{j \in T} x_{0,j}^k = \sum_{k \in U} \sum_{i \in T} x_{i,N+1}^k = V$$

$$C3: \sum_{j \in T \cup \{N+1\}} x_{i,j}^k \leq 1, \quad i \in T \cup \{0, N+1\}, \quad k \in U$$

$$C4: \sum_{i \in T \cup \{0, N+1\}} \sum_{j \in T \cup \{0, N+1\}} d_{i,j} \cdot x_{i,j}^k \leq D, \quad k \in U$$

$$C5: z_{i,j}^k - z_{i,j}^k + D \cdot x_{i,j}^k \leq D - d_{i,j}, \quad i \neq j \in T \cup \{0, N+1\}, \quad k \in U$$

$$C6: \sum_{j \in T \cup \{N+1\}} x_{i,j}^k = y_i^k, \quad i \in T, \quad k \in U$$  

(7)

where $C6$ defines the relationship of decision variables $x_{i,j}^k$ and $y_i^k$.

IV. METHODOLOGY

In this section, we introduce the DCF and our DL-DRL for solving the large-scale task scheduling problem of multi-UAV. Based on DCF, the original problem is decomposed into a task allocation subproblem and a UAV route planning subproblem, where the two subproblems are modelled in the upper level and lower level of the DCF, respectively. For the task allocation subproblem in the upper level, we first cast it as a 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 construct corresponding MDP and design a policy network to generate the route that maximizes the total value of tasks 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 the ITS that consists of pre-training, intensive training, and alternate training to achieve a balance between performance and training efficiency.
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 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.

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 total value of executed tasks 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 [51], OP [52], etc.

B. 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.

a) State: The state includes the current information of the UAV and task at each step, i.e., $s_t = (V_t, m_t) \in S$, where $V_t$ is the task set that have been allocated to UAVs at time step $t$, and $m_t$ is the task that need to be allocated at time step $t$. Particularly, $V_t = \{U_1^t, U_2^t, \ldots, U_n^t\} = \{u_1^t, u_2^t, \ldots, u_{m_t}^t\}$, where $U_i^t$ and $u_i^t$ are the task set and the corresponding task features (i.e., coordinates and value of the task) of the UAV $i$ at time step $t$, respectively.

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

c) 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., $s_{t+1} = (V_{t+1}, m_{t+1}) = P(s_t, a_t)$. The $m_{t+1}$ 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,

$$U_{i+1}^j = \begin{cases} U_i^j, m_i^j, & j = i \\ U_i^j, u_i^t, & \text{otherwise} \end{cases}$$

where $[ , ]$ refers to the concatenation operation, $m_i^j$ is the task allocated to UAV $i$ at time step $t$, and $u_i^t$ is the last element in $U_i^j$ (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.

d) Reward: Since we aim to maximize the task values that can be collected, we define the reward as the total values of tasks executed by all UAVs, i.e., $R = \sum v_i$, where $v_i$ is the task values collected by UAV $i$, which is obtained based on the planned route generated by the lower-level DRL model.

2) Architecture of the Policy Network: Compared to the conventional Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, the Transformer utilizes the self-attention mechanism to capture the relationships between any two elements within a sequence which can improve the feature extraction and overcome the constraint of sequence-aligned recurrence for the parallelization. As one of the most powerful architectures, it has been widely employed in domains of natural language processing [53], image recognition [54], recommendation systems [55], and combinatorial optimization [44]. Regarding the multi-UAV task scheduling problem, the input is a sequence of tasks (depot) characterized by locations and values. The construction of the task allocation scheme and UAV routes can be deemed as a sequential decision-making process. Thus, the Transformer-style policy network has the desirable potential to engender high-quality solutions with short computation time, and 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.

a) Encoder: The inputs of the encoder are the raw features of tasks and the depot, and we exploit a linear projection layer and multiple attention layers to learn more informative embeddings. Specifically, the raw inputs are first encoded to $d_e$-dimensional node embeddings $h^0$ via the linear projection with $d_e = 128$. 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 $h^{l-1} = [h_0^{l-1}, h_1^{l-1}, \ldots, h_N^{l-1}]$ 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 $h^{l-1}$
via the multi-head self-attention mechanism first. In specific, for each head, the vectors of query, key, and value are derived by multiplying $h^{l-1}$ and the corresponding trainable parameters. Then, we calculate the attention value $Y_{lm}$ according to Eq. (10), 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 $\hat{d}_l$ through the skip-connection and batch normalization operations. Afterwards, $\hat{d}_l$ 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,

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

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

$$\text{MHA}(h^{l-1}) = [Y_{l1}; \ldots; Y_{lM}] W_{lm}^O,$$

$$\hat{h}_l = BN(h^{l-1} + \text{MHA}(h^{l-1})),$$

$$h_l = BN(\hat{h}_l + \text{FF}(\hat{h}_l)),$$ (13)

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

b) Decoder: The output of the encoder $h^k$ 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 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., $\tilde{C}_i = [h_{0i}^{L_i}, h_{1i}^{L_i}, \ldots, h_{j_i}^{L_i}, \ldots, h_{J_i}^{L_i}]$, where $h_{ji}^{L_i}$ 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 $\tilde{C}_iT_S$. After that, a linear projection and a 512-dimensional FF layer are used to yield the task subset embeddings $H_i^{T_S}$. Finally, we concatenate $H_i^{T_S}$ and node embedding of the current task $h_t^i$, and then feed it to a linear projection and the softmax function to obtain 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 former 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,

$$\tilde{C}_i = \max(\tilde{C}_i),$$

$$\tilde{C}_iT_S = [\tilde{C}_i^1, \tilde{C}_i^2, \ldots, \tilde{C}_i^N],$$

$$H_i^{T_S} = FF(W_i \tilde{C}_iT_S + b_1),$$

$$p_i = \text{softmax}(W_i [H_i^{T_S}, h_t^i] + b_2),$$

where $W_i, b_1, W_2$ and $b_2$ are trainable parameters. In this way, the upper-level model functions as a “classifier” that divides tasks into different UAVs. Consequently, the corresponding models need to be trained offline from scratch for scenarios with different number of UAVs.

C. 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 [28]
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. Thus, we set the reward of the UAV route planning subproblem as the value of tasks executed by a UAV, and the definition of the MDP is stated as follows.

1) State: The state is the current information of the tasks and UAV at each step, i.e., $s_i = (U'_i, v_i)$, where $U'_i$ is the task set that allocated to the UAV and $v_i$ contains the position $p_i$ and remaining flight distance $f_i$ of the UAV at time step $t$, i.e., $v_i = (p_i, f_i)$. Particularly, $U'_i = \{m_1^i, m_2^i, \ldots, m_n^i\} = \{(p'_1, r'_1), (p'_2, r'_2), \ldots, (p'_n, r'_n)\}$, where $p'_i$ is the position of the allocated task $i$ and $r'_i$ is the value of task $i$ ($r'_i$ will become 0 once that task has been executed). Here, only tasks with $r'_i > 0$ can be executed.

2) Action: The action is to determine the next task to execute, i.e., $a'_i = m'_i$, where $m'_i = (p'_i, r'_i)$ is the next task need to be executed by the UAV.

3) State Transition Rule: The state transition rule will transit the previous state $s'_i$ to the next state $s'_{i+1}$ based on the performed action $a'_i$ i.e., $s'_{i+1} = (U'_{i+1}, v_{i+1}) = P'(s'_i, a'_i)$, where $v_{i+1} = (p'_i, f_i - \text{distance}(p_i, p'_i))$.

4) Reward: The reward is set to the total value of tasks executed by the UAV.

Similar to AM, the encoder-decoder architecture is used as the policy network for route construction. In the encoder, the 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,

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

where $D_t$ is the allowed remaining flight distance at time step $t$, $d_{\pi_t, \pi_{t+1}}$ is the distance between task $\pi_t$ and $\pi_{t+1}$, and $\pi_0$ 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 [41] 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,

$$C^i_t = [h_G, h_{t-1}, D_t],$$

$$\tilde{C}^t = \text{glimpse}(C^t),$$

$$u^i_t = C \cdot \tanh(\frac{q^T_{li} k^i_t}{\sqrt{\text{dim}_k}}),$$

$$p^i_t = \text{softmax}(u^i_t),$$

where $u^i_t$ is the compatibilities calculated by a single-head attention operator; $q^T_{li} = \tilde{C}^T_i W^q_{\text{com}}$, and $k^i_t = h^N W^k_{\text{com}}$ are the query and key vectors for the attention operation, with $h^N$ being the embedding of the tasks, $W^q_{\text{com}}$ and $W^k_{\text{com}}$ being the trainable parameters; $p^i_t$ 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.

Based on the constructed upper-level and lower-level DRL models for task allocation and route planning, we can obtain the scheduling scheme (i.e., the task execution order for each UAV) of the input tasks and depot. While our architecture of the policy network is tailored to the specific context of multi-UAV task scheduling, it can adapt and extend to more generic situations with a slight adjustment. For example, by changing the decoder context of the lower-level DRL model for solving TSP [28], our DL-DRL can be used to solve m-TSP, where the upper-level model is used to assign customers to certain agents and the lower-level model plans the visiting route for each agent. Moreover, based on the divide-and-conquer principle, we can decompose the complex problem into several simple subproblems, and corresponding DRL models can be constructed and trained for solving the subproblems. In this way, the idea of our approach can also adapt and extend to other complex problems.

D. 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 the 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 [44, 47, 56]. Accordingly, the gradient of the loss function in our two models is calculated as follows,

$$\nabla L(\theta | s) = E_{p_{\theta}(\pi | s)}[(L(\pi) - L(\pi^{BL})) \nabla \log p_{\theta}(\pi | s)],$$

where $\theta$ is the parameters of the policy network $\pi$, $L(\pi)$ and $L(\pi^{BL})$ are the cost function (i.e., the negative value 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.

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 $E_t$ 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.

Based on ITS, our policy networks can learn useful patterns from massive data during training that can generalize to unseen instances and have the potential to solve similar problems rather than designing problem-specific rules, such as the problem of natural disaster rescue that desires to find more survivors by dispatching vehicles efficiently, the problem of traffic reconnaissance that expects to maximize the benefit of reconnaissance by using multiple vehicles.

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 attention layer, DRL modules and ITS are also assessed by corresponding ablation study. Besides, we further test our DL-DRL on instances of larger-scale and different flight distances of UAVs to verify its generalization performance.

A. Experiment Settings

Following the convention in [28], [56], and [57], 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 and the value of all tasks is set to 1.0 as [5] and [6] 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 gradient 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.
TABLE II
EPOCH TIME OF TRAINING OUR MODELS IN VARIOUS SITUATIONS (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          |

B. Learning Performance

Following the above experiment settings, our DL-DRL model is trained on U4 and U6 scenarios with different task 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. In our DL-DRL, the training time consumption comprises the training of the upper-level model, the training of the lower-level model, and the data generation for the lower-level model. Among these three parts of time consumption, the training of the lower-level model takes the shortest time, whereas the training of the upper-level model takes the most time as listed in Table II. In particular, “U4-80” means the model is trained on the U4 scenario with 80 tasks.

We can observe that the training time consumption of our DL-DRL is affected by two factors, i.e., the task size and the number of UAVs. With the increase in task size, the input data dimension for both the upper-level and lower-level models grows, resulting in a proportional increase in the training time for both models. Additionally, the data generation time for the lower-level model also experiences a corresponding increase. As the number of UAVs affects both the decision-making process within the upper-level model and the data generation for the lower-level model, the increase in the number of UAVs leads to more time consumption for both the training of upper-level model and data generation. However, since the route planning for a single UAV in the lower-level models is similar across situations with varying UAV quantities, the training time for the lower-level model remains relatively consistent if the task scale is the same. In addition, 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.

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 value 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\footnote{https://www.gurobi.com/}: A commercial exact solver for (mixed) integer programmings;
2) OR-Tools\footnote{https://developers.google.cn/optimization/}: A widely used strong heuristic solver developed by Google;
3) K-means+VND: Based on the same DCF, it uses K-means and variable neighborhood descent (VND)\(^3\) to handle task allocation and UAV route planning, respectively;

4) K-means+AM: Based on the same DCF, it uses K-means and AM [28] to handle task allocation and UAV route planning, respectively;

5) DL-DRL-I: An approach utilizes the same network architecture as our DL-DRL, while it trains the two DRL models independently rather than using the proposed ITS. More information about DL-DRL-I is in the supplementary.

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. (17), 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 III and Table IV, 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 value of executed tasks on average, and the gap is defined as the normalized distance between an average objective value \(\text{Obj}\) and the best one \(\text{Obj}_{\text{best}}\) found among all methods, which is expressed as follows,

\[
\text{Gap} = \frac{\text{Obj}_{\text{best}} - \text{Obj}}{\text{Obj}_{\text{best}}} \times 100%.
\]  (24)

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 III and IV 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 III, 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.

\(^3\)https://github.com/nchlpz/VND _TOP
Pertaining to the U6 scenario in Table IV, 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.

Furthermore, to investigate assigned results generated by approaches based on DCF (i.e., K-means + VND, K-means + AM, DL-DRL-I, and DL-DRL), we visualize the task allocation schemes engendered by K-means, the upper-level model of DL-DRL-I, and the upper-level model of DL-DRL on U4 and U6 scenarios with 80 tasks in Fig. 5. In particular, the depot and tasks are presented as a square and dots, respectively, and tasks with the same color are allocated to the same UAV.
K-means (Fig. 5(a) and 5(d)) groups tasks solely based on their spatial locations, forms proximate tasks into a cluster and allocates them to a UAV. However, due to the limit of UAV flight distance, few tasks in the clusters that are far away from the depot might be completed. For the upper-level model in DL-DRL-I (Fig. 5(b) and 5(e)), we can obviously see that the allocation of tasks in the left bottom is unreasonable as these tasks are remote from other tasks that have been allocated to the same UAV. In contrast, tasks are allocated rationally for the upper-level model in DL-DRL (Fig. 5(c) and 5(f)), which takes into account both the locations of tasks and the depot as well as the limitation of UAV flight distance, allowing multi-UAV to complete more tasks. Moreover, we visualize the scheduling results of K-means + AM, DL-DRL-I, and our DL-DRL in Fig. 6, where the UAV routes planned by our DL-DRL have no apparent intersections and can execute more tasks.

D. Ablation Study on Attention Layer

The attention layer is constructed on both policy networks of the upper-level and lower-level DRL models in our DL-DRL and is utilized to focus on different parts of the input sequence (i.e., the initial embeddings of raw features, including the depot and tasks), selectively attending to relevant information during the task scheduling process for multi-UAV. By computing attention values, the attention layer would help the DRL model weigh the importance of different tasks in relation to each other, which enhances its ability to capture dependencies among input information. To investigate the influence of the attention layer, we conduct an ablation study on the attention layer in the U4 scenario with 80 tasks, where the learning curves of DL-DRL models with and without the presence of the attention layer are shown in Fig. 7. Particularly, “DL-DRL/Atten_Layer” denotes the attention layers absent on both policy networks of the upper-level and lower-level DRL models. “Upper/Atten_Layer” and “Lower/Atten_Layer” denote the attention layers absent on the policy networks of the upper-level DRL model and the lower-level DRL model, respectively.

According to Fig. 7, our DL-DRL achieves the best learning performance in both upper-level and lower-level models, which verifies the absence of the attention layer significantly degrades the learning performance of the DL-DRL. Comparing the influence of the attention layers in upper-level and lower-level models, we can find that the absence of the attention layer in the lower-level model (i.e., “Lower/Atten_Layer”) would lead to worse learning performance than that in the upper-level model (i.e., “Upper/Atten_Layer”), which demonstrates the attention layer has a more pronounced impact in UAV routing subproblem than task allocation subproblem. Thus, we conclude that the attention layer plays a significant role in the performance of both upper-level and lower-level models, and thereby is critical for the overall performance of our DL-DRL.

E. Ablation Study on DRL Modules

To illustrate the effectiveness of each DRL modules in our approach (i.e., the upper-level DRL module and the lower-level DRL module), we conduct an ablation study by replacing the upper-level and lower-level DRL modules with a heuristic algorithm SATL [17] and OR-Tools, respectively.
We can observe that our DL-DRL outperforms DRL+OR-Tools and SATL+DRL in terms of both Obj. and computation time, demonstrating each DRL module in our method plays an important role in the task scheduling of multi-UAV. On the other hand, the computation time of DRL+OR-Tools and SATL+DRL is significantly longer than that of DL-DRL, showing the efficiency of our DL-DRL for the problem solving.

**F. Ablation Study on ITS**

To investigate the efficacy of different processes in ITS, we construct an ablation study on the scenario U4 with 80 tasks. Different combinations of the training strategies are used, and we record the cost value during the training process in Fig. 8. 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. 8, 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 II, 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.

G. 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. 9, 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. 9(a), 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$+\text{VND}$, K-means$+\text{AM}$, 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. 9(b), where the models trained for other problem sizes outperform K-means$+\text{VND}$ and K-means$+\text{AM}$ and are competitive against DL-DRL-I.

For larger-scale multi-UAV task scheduling problems with more than 1000 tasks, it is intractable to train the model from scratch, rendering 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, 1000, and 1500 tasks, and also compare it with the baselines including OR-Tools, K-means$+\text{AM}$, and DL-DRL-I, where the sampling strategy is used for both DL-DRL and DL-DRL-I.

Table VII and Table VIII list the testing results for the U4 and U6 scenarios, including the total value of executed tasks on average (Obj.) and the average computation time (Times), respectively. As can be seen, our DL-DRL outperforms K-means$+\text{VND}$ and K-means$+\text{AM}$ and are competitive against DL-DRL-I. Comparing with OR-Tools, DL-DRL shows similar effectiveness on small-scale instances (i.e., $N=80$ and $N=100$) with less computation time, whereas it outperforms OR-Tools on large-scale instances (i.e., $N=300$ and $N=500$) in terms of both Obj. and computation time. The results demonstrate that our DL-DRL has strong generalization ability on the different flight distances of UAVs, especially for large-scale instances.

### 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 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 ITS and two DRL modules are also demonstrated via ablation studies. Regarding the future studies, we will further improve our approach to tackle various distributions of task values, generalize to a different number of UAVs, test on real-world instances, and consider diverse dynamic scheduling situations, such as tasks sudden appearance or disappearance and UAV damage.

REFERENCES

[1] Y. Liu, “An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones,” Comput. Oper. Res., 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, Switzerland: Springer, 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,” Mechatronics, vol. 31, no. 2, pp. 339–350, Feb. 2018.

[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,” Chin. J. Aeronaut., vol. 31, no. 2, pp. 393–398, Feb. 2018.

[5] J. Turner, Q. Meng, G. Schaefer, A. Whitbrook, and A. Soltoggio, “Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system,” IEEE Trans. Cybern., vol. 48, no. 9, pp. 2583–2597, Sep. 2018.

[6] 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 Intell. Syst., vol. 7, no. 2, pp. 873–890, Apr. 2021.

[7] L. Zuo, S. Gao, Y. Li, L. Li, M. Li, and X. Lu, “A fast and robust algorithm with reinforcement learning for large UAV cluster mission planning,” Remote Sens., vol. 14, no. 6, p. 1304, Mar. 2022.

[8] G. Laporte and Y. Nobert, “A cutting planes algorithm for the m-Salesmen problem,” J. Oper. Res. Soc., vol. 31, no. 11, pp. 1017–1023, Nov. 1980.

[9] P. B. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multirobot systems,” Int. J. Robot. Res., vol. 23, no. 9, pp. 939–954, Sep. 2004.

[10] A. Richards, J. Bellingham, M. Tillerson, and J. How, “Coordination and control of multiple UAVs,” in Proc. AIAA Guid., Navigat., Control Conf. Exhibit, Aug. 2002, p. 4588.

[11] M. Alighanbari and J. How, “Cooperative task assignment of unmanned aerial vehicles in adversarial environments,” in Proc. Amer. Control Conf., 2005, pp. 4661–4666.

[12] E. J. Forssø, E. I. Grotti, T. I. Fossen, and T. A. Johansen, “Optimal search mission with unmanned aerial vehicles using mixed integer linear programming,” in Proc. Int. Conf. Unmanned Airc. Syst. (ICUAS), May 2013, pp. 253–259.

[13] E. Edison and T. Shimha, “Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms,” Comput. Oper. Res., vol. 38, no. 1, pp. 340–356, Jan. 2011.

[14] S. Gao, J. Wu, and J. Ai, “Multi-UAV reconnaissance task allocation for heterogeneous targets using grouping ant colony optimization algorithm,” Soft Comput., vol. 25, no. 10, pp. 7155–7167, May 2021.
S. Luo, L. Zhang, and Y. Fan, “Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning,” IEEE Trans. Autom. Sci. Eng., vol. 19, no. 4, pp. 3020–3038, Oct. 2022.

Y. Wang, S. Sun, and W. Li, “Hierarchical reinforcement learning for vehicle routing problems with time windows,” in Proc. Can. Conf. Artif. Intell., Jun. 2021, pp. 1–6.

I. Bello, H. Pham, Q. Le, M. Norouzi, and S. Bengio, “Neural combinatorial optimization with reinforcement learning,” Feb. 2017, arXiv:1611.09940.

O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 28, 2015, pp. 2692–2700.

Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, “Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning,” 2019, arXiv:1911.04936.

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 Trans. Cybern., vol. 52, no. 10, pp. 11107–11120, Oct. 2022.

L. Zhao, J. Fan, C. Zhang, W. Shen, and J. Zhuang, “A DRL-based reactive scheduling policy for flexible job shops with random job arrivals,” IEEE Trans. Autom. Sci. Eng., early access, pp. 1–12, May 2023.

F. Luo, X. Lin, F. Liu, Q. Zhang, and Z. Wang, “Neural combinatorial optimization with heavy decoder: Toward large scale generalization,” in Proc. 37th Conf. Neural Inf. Process. Syst., 2023, pp. 1–20.

Y. Hu, Y. Yao, and W. S. Lee, “A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs,” Knowl.-Based Syst., vol. 204, Sep. 2020, Art. no. 106244.

Q. Hou, J. Yang, Y. Su, X. Wang, and Y. Deng, “Generalize learned heuristics to solve large-scale vehicle routing problems in real-time,” in Proc. 11th Int. Conf. Learn. Represent., 2023, pp. 1–37.

J. Chen et al., “Global-and-local attention-based reinforcement learning for cooperative behaviour control of multiple UAVs,” IEEE Trans. Veh. Technol., early access, pp. 1–13, Oct. 2023.

C. E. Miller, A. W. Tucker, and R. A. Zemlin, “Integer programming formulation of traveling salesman problems,” J. ACM, vol. 7, no. 4, pp. 326–329, Oct. 1960.

X. Xu, H. Yuan, M. Liptrott, and M. Trovati, “Two phase heuristic algorithm for the multiple-travelling salesman problem,” Soft Comput., vol. 22, no. 19, pp. 6567–6581, Oct. 2018.

I. Dolinskaya, Z. E. Shi, and K. Smilowitz, “Adaptive orienteering problem with stochastic travel times,” Transp. Res. E. Logistics Transp. Rev., vol. 109, pp. 1–19, Jan. 2018.

I. V. Tetro, P. Karpov, R. Van Deursen, and G. Godin, “State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis,” Nature Commun., vol. 11, no. 1, p. 5575, Nov. 2020.

A. Dosovitskiy et al., “An image is worth 16 × 16 words: Transformers for image recognition at scale,” in Proc. Int. Conf. Learn. Represent., Sep. 2020, pp. 1–21.

F. Sun et al., “BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., Nov. 2019, pp. 1441–1450.

J. Li et al., “Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem,” IEEE Trans. Cybern., vol. 52, no. 12, pp. 13572–13585, Dec. 2022.

M. Nazari, A. Oroojlooy, L. Snyder, and M. Takie, “Reinforcement learning for solving the vehicle routing problem,” in Proc. Adv. Neural Inf. Process. Syst., vol. 31, 2018, pp. 9839–9849.

**Guohua Wu** (Senior Member, IEEE) received the B.S. degree in information systems and the Ph.D. degree in operations research from the National University of Defense Technology, China, in 2008 and 2014, respectively. From 2012 to 2014, he was a Visiting Ph.D. Student with the University of Alberta, Edmonton, Canada. He is currently a Professor with the School of Traffic and Transportation Engineering, Central South University, Changsha, China. He has authored more than 100 refereed articles, including those published in IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. His current research interests include scheduling, computational intelligence, and machine learning.

He serves as an Associate Editor for Information Sciences and Swarm and Evolutionary Computation, an Editorial Board Member for International Journal of Bio-Inspired Computation, and a Guest Editor for Information Sciences and Memetic Computing.

**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.

**Zhiguang Cao** (Member, IEEE) received the B.Eng. degree in automation from the Guangdong University of Technology, Guangzhou, China, the M.Sc. degree in signal processing from Nanyang Technological University, Singapore, and the Ph.D. degree from the Interdisciplinary Graduate School, Nanyang Technological University. 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 joined the School of Computing and Information Systems, Singapore Management University, as an Assistant Professor. His research interests focus on learning to optimize (L2Opt).

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

**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 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. 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 articles in the above areas.

In 2012, he was elected as a fellow of the Royal Society of Canada. He is the Editor-in-Chief of Information Sciences. He currently serves as an Associate Editor for 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.