Swarm and Evolutionary Computation 93 (2025) 101858

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Multi-UAV reconnaissance mission planning via deep reinforcement learning with simulated annealing☆

Mingfeng Fan [a], Huan Liu [b], Guohua Wu [c,*], Aldy Gunawan [d], Guillaume Sartoretti [a]

[a] Department of Mechanical Engineering, National University of Singapore, Singapore
[b] School of Traffic and Transportation Engineering, Central South University, Changsha, China
[c] School of Automation, Central South University, Changsha, China
[d] School of Computing and Information Systems, Singapore Management University, Singapore

ARTICLE INFO

Keywords:
Multi-UAV
Deep reinforcement learning
Allocation and routing
Simulated annealing

ABSTRACT

Unmanned aerial vehicles (UAVs) are widely used in reconnaissance missions due to their autonomy and flexibility. Efficient mission planning for multiple UAVs is crucial for tasks such as traffic monitoring and data collection. However, existing approaches to multi-UAV reconnaissance mission planning problem (MURMPP) often struggle with high computational demands, leading to suboptimal solutions. To overcome this challenge, we introduce a divide-and-conquer framework that splits the problem into two phases: target allocation and UAV routing, effectively reducing computational complexity. Specifically, we propose a hybrid method, SA-NNO-DRL, which combines the nearest neighbor optima-based deep reinforcement learning (NNO-DRL) approach with simulated annealing (SA). In the UAV routing phase, NNO-DRL constructs routes for each UAV, while SA reassigns uncovered targets during the target allocation phase. The two phases alternate until the termination condition is met. Experimental results show that our method outperforms exact solvers, heuristics, and learning-based approaches, finding the most solutions deemed best in 8 out of 12 instance groups within 0.5 s. Our method particularly excels in larger problems and adapts well to varying target sizes, hub locations, and UAV numbers.

1. Introduction

With the rapid urbanization and surge in vehicle usage in recent years, the transportation industry is posing increasingly high demands in information collection about traffic state and road condition [1]. As a modern and efficient alternative to traffic monitoring, unmanned aerial vehicles (UAVs) exhibit unique advantages due to the desired accuracy, flexibility, and relatively low cost in information capture and transmission [2,3]. Typically, UAVs collect image or video data at some key road junctions and then transmit them to traffic management centers for further processing, based on which the respective agency could detect traffic accidents, identify traffic jams, check traffic infrastructures, etc. Besides, UAVs are preferable also in that they can travel freely over an urban area with scattered obstacles [4], which move much faster than ground vehicles. Compared to an individual, multiple UAVs could significantly expand monitoring areas and fulfill more challenging missions by performing numerous tasks concurrently and cooperatively, the performance of which is usually contingent on the underlying mission planning.

In this paper, we study the multi-UAV reconnaissance mission planning problem (MURMPP) [5] in the context of traffic monitoring, *i.e.*, a cooperative combinatorial optimization problem in essence, which is hard to solve. The problem involves dispatching multiple UAVs to monitor specific targets (nodes) in a given area. Each target can only be covered by one UAV at most. However, UAVs face constraints, such as limited battery capacity, which restricts their flying range. This limits how far each UAV can travel to monitor distant targets. Additionally, the number of UAVs is often insufficient to cover all the targets in the area, meaning some targets may remain unmonitored. Each target is assigned a profit value, reflecting its priority or the importance of monitoring it in the traffic monitoring system. For instance, in the case of traffic congestion, profits can be determined based on historical or real-time traffic data, allowing UAVs to prioritize monitoring congested areas and gathering more valuable information for traffic management [6]. Accordingly, MURMPP aims to maximize the total profits w.r.t the resulting solution by properly planning the mission for the given group of UAVs. Note that studying MURMPP also benefits practical applications of other domains, such as disaster rescue [7]

☆ This work was supported by the National Natural Science Foundation of China (Grant No. 62373380).
* Corresponding author.
E-mail address: guohuawu@csu.edu.cn (G. Wu).

https://doi.org/10.1016/j.swevo.2025.101858
Received 27 August 2024; Received in revised form 10 December 2024; Accepted 14 January 2025
Available online 28 January 2025

and customer selection in less-than-truckload transportation [8], where efficient and effective mission planning is crucial.

MURMPP can be viewed as a team orienteering problem (TOP) and prior works mainly tackle it as a whole. Several exact approaches [9–11] have been developed for TOP but can only handle up to 100 nodes (targets) due to the curse of dimensionality. Classic heuristic (or meta-heuristic) algorithms could attain satisfactory solutions in an acceptable time, such as ant colony optimization (ACO) [12], tabu search (TS) [13,14], particle swarm optimization (PSO) [15] and simulated annealing (SA) [16]. Nevertheless, the performance of heuristic algorithms considerably hinges on the designed operators or rules, and the high computation complexity prevents them from being applied to large-scale problems [17]. On the other hand, exploiting deep reinforcement learning (DRL) for combinatorial optimization problems has been receiving growing attention recently, like vehicle routing problem [18–22], knapsack problem [23] and job shop scheduling problem [24,25]. However, rare attempts have been made for MURMPP given two foreseeable challenges if DRL is directly used to tackle the whole MURMPP, *i.e.*, (1) the search space increases exponentially with the number of UAVs or targets; (2) the cooperative relationship among UAVs is so complex that the globally optimal reward (*i.e.*, the total profits) is hard to achieve.

To mitigate the computational overhead, we split the MURMPP into two phases with the divide-and-conquer principle, *i.e.*, allocating targets for the given group of UAVs and routing for the respective UAV. Inspired by the competitive performance and broad success achieved by hybrid models that combine machine learning and operation research (OR) algorithms [17,26,27], we propose to leverage DRL and meta-heuristic to handle the two phases for solving MURMPP, respectively. Given the initially allocated targets, we present a DRL algorithm to construct a route for each UAV and reassign the uncovered targets using a well-designed SA algorithm. We tackle the two subproblems alternately and cooperatively rather than solving them independently without interaction. This method leverages the respective strength of machine learning and OR, where DRL can tackle the routing problem for an individual UAV quickly and meta-heuristic can induce high total profits in a global view.

Specifically, we first evenly divide targets into several groups, with each for a UAV. Then, we exploit the nearest neighbor optima-based DRL model (NNO-DRL) to construct the path for each UAV, which is expected to pass through the group of the assigned target (nodes) for the monitoring task. However, the maximum flight range typically restricts UAVs, and some UAVs may not be able to monitor all targets assigned to them. In this case, we employ the SA algorithm to perform a directed shift operation, feeding the uncovered targets back to the target allocation phase and reassigning them to other UAVs. We perform the above target allocation and UAV routing iteratively until the predefined termination condition is met. Notably, the proposed two-phase framework can be readily generalized to MURMPPs with different numbers of UAVs and targets since we decompose the routing for multi-UAV into individual ones and coordinate them through the target allocation. In doing so, we also save the overhead of training separate DRL models and accelerate the inference by routing all UAVs parallel in each iteration. Furthermore, our method using the DRL model follows the offline learning paradigm and can solve a class of MURMPP instances once the training is done. Accordingly, the main contributions of this paper are outlined as follows,

1. We propose an iterative two-phase framework, *i.e.*, SA-NNO-DRL, which decomposes the MURMPP into a target allocation subproblem and a UAV routing subproblem. It effectively reduces the problem's complexity and generalizes well to different numbers of targets and/or UAVs without re-training.
2. We introduce an NNO-DRL model to solve routing subproblems for UAVs, which samples multiple trajectories with first targets specified as the k-nearest neighbors of the UAV hub. Additionally, we propose a joint training method that combines a contrastive learning (CL) strategy and the REINFORCE algorithm to effectively train our NNO-DRL model. We also design a SA algorithm with a directed shift operation in the local search, which is specialized for target reallocation.
3. We conduct extensive experiments to evaluate our method, confirming its superiority to other baselines in terms of solution quality and generalization performance.

The remainder of the paper is organized as follows. We review the related works in Section 2. We present the mathematical formulation of MURMPP in Section 3. We elaborate the proposed method in Section 4. Our experiments and analysis are presented in Section 5, and finally, we conclude the paper in Section 6.

2. Related work

As a cooperative combinatorial optimization problem in nature, MURMPP aims to achieve globally optimal profits by coordinating the respective agents (UAVs). Existing methods for MURMPPs mainly encompass two categories, *i.e.*, solving it wholly with a single optimization approach or decomposing it into subproblems in a hierarchical manner.

2.1. Existing methods for MURMPPs

On the one hand, MURMPP is usually formulated as a mixed-integer linear program (MILP) or Markov decision process (MDP) in a multi-agent system. Forsmo et al. [28] proposed a MILP model to plan a joint search mission for multiple UAVs. However, yielding a complete solution for all UAVs in each iteration is inefficient. Cui et al. [29] proposed a multi-agent reinforcement learning method to realize the autonomous resource allocation for multiple UAVs. Sankaran et al. [30] proposed a graph attention model for multiple agents (GAMMA) to address a multi-agent reconnaissance mission planning problem. However, the spaces of both state and action exponentially increase with the number of agents in multi-agent learning [31], which limits such methods to only small-sized problems. In contrast, hierarchical approaches are less explored. Yao et al. [32] proposed a distributed mission planning framework with separate modules for task assignment and path planning. Liu et al. [33] divided multi-UAV scheduling into target allocation and individual UAV scheduling subproblems. Mao et al. [34] similarly decomposed the multi-UAV scheduling problem, applying a double-layer DRL method to address the subproblems. While efficient, this approach struggles to generalize to varying numbers of UAVs. MURMPP generally comprises two primary combinatorial optimization subproblems, *i.e.*, target allocation and UAV routing. Due to their NP-hardness, achieving efficient and effective solutions for both remains challenging. To address this, we propose an NNO-DRL method for fast UAV routing and an SA algorithm for global task reallocation.

2.2. Meta-heuristics for task allocation

Exact and heuristic algorithms are two primary types of methods for tackling the task allocation problem. While exact algorithms such as dynamic programming [35], and branch and bound [36] have the potential to attain the optimal solution, they suffer from intractable computation for large-scale problems. In contrast, heuristics are more commonly used as they usually attain good approximate solutions in a reasonable time. Li et al. [37] adopted an ACO method to handle the task allocation, in which multiple UAVs are scheduled to attack the ground targets given the constraint of flight range. Tian et al. [38] proposed a genetic algorithm (GA) to perform multiple reconnaissance tasks for UAVs with specialized crossover and mutation operators. Bai et al. [39] combined the GA with tabu search (TS) to improve the efficiency of task allocation for multiple vehicles. In addition, a self-organizing map (SOM) based method was proposed to deal with a

multi-task planning problem, the efficiency of which decreases as the problem scale grows [40]. Wu et al. [41] proposed a hybrid genetic and simulated annealing algorithm for task allocation in multiple UAV formations. Xu et al. [42] introduced a lazy-based review consensus algorithm for multi-vehicle task assignment, enabling each vehicle to compute its allocation independently while resolving conflicts through a consensus strategy. They further developed a lazy auction strategy to accelerate convergence [43]. Unlike previous works focused on local or population-based strategies, our SA algorithm is designed to efficiently handle task reallocation in MURMPPs, iteratively refining task assignments to balance solution quality with computational cost.

2.3. DRL for routing

DRL models can learn useful patterns from extensive data without relying on complex heuristics or domain-specific knowledge, which can generalize to unseen instances [44]. Furthermore, trained DRL models are very efficient during inference. Upon these advantages, there has been a surge of research to apply DRL for solving routing problems, such as the classic traveling salesman problem (TSP) and capacitated vehicle routing problem (CVRP) in recent years. According to the solution generation scheme, the DRL-based routing methods are generally classified into two different fashions, *i.e.*, *improvement* [19,22,45–49] and *construction* [18,20,21,50–54]. The constructive DRL-based routing methods generally have a faster inference time than the improvement ones [55], which is exactly what we seek for solving MURMPP. In specific, Bello et al. [50] first proposed one of the earliest DRL-based routing methods for TSP by training a pointer network (PtrNet) [56] with an actor-critic algorithm. Since positional information was not meaningful for routing problems, Nazari et al. [51] replaced the long short-term memory (LSTM) encoder in PtrNet with element-wise projection for solving CVRP. Then, Kool et al. [18] proposed an attention model(AM) to generate higher-quality solutions, where a Transformer-based architecture [57] replaced PrtNet. Xin et al. [21] extended AM by explicitly removing the visited nodes in each node selection step. The multi-decoder AM (MDAM) [20] utilized a Transformer for node encoding and multiple attention decoders with separate parameters to sample diverse trajectories. These constructive DRL-based routing methods mainly employ a START token, *i.e.*, trainable parameters, as the input to the decoder to decide the first node (target) in a route. However, such parameters are inherently difficult to train, potentially resulting in a suboptimal search space for subsequent actions due to symmetries in the routing problem solutions. The policy optimization for multiple optimal (POMO) [52] leveraged parallel solution trajectories and employed data augmentation techniques to account for the symmetrical nature of routing solutions, achieving significantly improved performance compared to AM. Building upon POMO, Kim et al. [53] further exploited problem symmetry to enhance its performance. Recent studies have also explored alternative approaches to solving VRPs, such as learning heuristics for ACO [58,59] and the divide-and-conquer method [60–62]. Building on these approaches, our NNO-DRL method explores a projection multi-layer perception(MLP) and CL for advanced problem representation, and a multi-start decoder with a k-nearest neighbors strategy to explore the solution space.

Note that a rare attempt has been made to leverage DRL and meta-heuristics for MURMPP cooperatively. In this paper, we present an iterative two-phase framework that integrates DRL and SA, where assigning targets is fulfilled by SA in high-level while DRL is exploited for respective UAV routing in low-level.

3. Problem definition and formulation

The goal of MURMPP is to find paths for a given group of UAVs that maximize the total profits collected by them for monitoring targets. In MURMPP, each UAV departs from the same hub and returns to it after the monitoring task. Note that successful monitoring here refers to visiting or passing through a target (node) by the respective UAV. The profits attached to targets are predefined by an underlying distribution, and the UAV collects the profit only after successfully monitoring a target. In addition, some of the targets might not be covered by UAVs due to the flight range constrained by the battery. The MURMPP is essentially a complex combinatorial optimization problem with a discrete solution space, and its intractability grows exponentially as the number of targets rises.

Without the loss of generality, we assume that there are m UAVs and n targets, where the UAV hub is denoted by 0, the UAV set is represented as $M = \{1, 2, \dots, m\}$ and the target set is represented as $N = \{1, 2, \dots, n\}$. A node set $I = \{0, 1, \dots, n\}$ includes all nodes (*i.e.*, targets and the UAV hub), in which each node i is attached with a profit p_i and $p_0 = 0$. In particular, $x_{i,j,b} = 1$ denotes that node j is visited after node i by UAV b, otherwise $x_{i,j,b} = 0$; $y_{i,b} = 1$ represents that node i is visited by UAV b, otherwise $y_{i,b} = 0$; $z_{i,b}$ is a auxiliary real-value variable used for subtour elimination; $d_{i,j}$ denotes the distance between node i and node j; L_{max} denotes the maximum flight range. Accordingly, the MURMPP could be formulated as a MILP as follows,

$$\max \quad \sum_{i\in I}\sum_{j\in N}\sum_{b\in M} p_j x_{i,j,b} \tag{1}$$

$$\text{s.t.} \quad \sum_{b\in M} y_{i,b} \le 1, \forall i \in N, \tag{2}$$

$$\sum_{b\in M} y_{0,b} = m, \tag{3}$$

$$\sum_{j\in I} x_{i,j,b} - \sum_{j\in I} x_{j,i,b} = 0, \forall i \in I, b \in M \tag{4}$$

$$\sum_{i\in I} x_{i,j,b} = y_{j,b}, \forall j \in I, b \in M \tag{5}$$

$$\sum_{j\in I} x_{i,j,b} = y_{i,b}, \forall i \in I, b \in M, \tag{6}$$

$$\sum_{i\in I}\sum_{j\in I} d_{i,j} x_{i,j,b} \le L_{max}, \forall b \in M, \tag{7}$$

$$z_{i,b} - z_{j,b} + L_{max} x_{i,j,b} \le L_{max} - d_{i,j}, \quad \forall i \in N, j \in N, b \in M, i \ne j, \tag{8}$$

$$z_{i,b} \ge 0, \forall i \in I, b \in M, \tag{9}$$

$$z_{i,b} \le L_{max} - d_{i,0}, \forall i \in I, b \in M, \tag{10}$$

$$x_{i,j,b} = \{0, 1\}, \forall i \in I, j \in I, b \in M, i \ne j. \tag{11}$$

The above MILP aims to maximize the total profits collected by UAVs, which is described by the objective function in an expression (1). Inequality (2) guarantees that each target is monitored (or visited) by one UAV at most. Eq. (3) ensures that all UAVs start from and return to the same UAV hub. Eqs. (4), (5) and (6) ensure that the node i visited by UAV b has exactly one preceding node and one succeeding node. Inequality (7) constrains that the total distance traveled by a UAV is within its maximum flight range. Constraints (8), (9) and (10) are used for subtour elimination. Finally, Eq. (11) specifies the domain of the binary variable $x_{i,j,b}$.

4. SA-NNO-DRL method

To solve the MURMPP, we present an iterative two-phase framework, *i.e*, SA-NNO-DRL, where the target allocation and UAV routing are alternately executed and mutually interacted. In particular, we first introduce the NNO-DRL routing method for individual UAVs and then elaborate the SA method specifically designed for the target allocation.

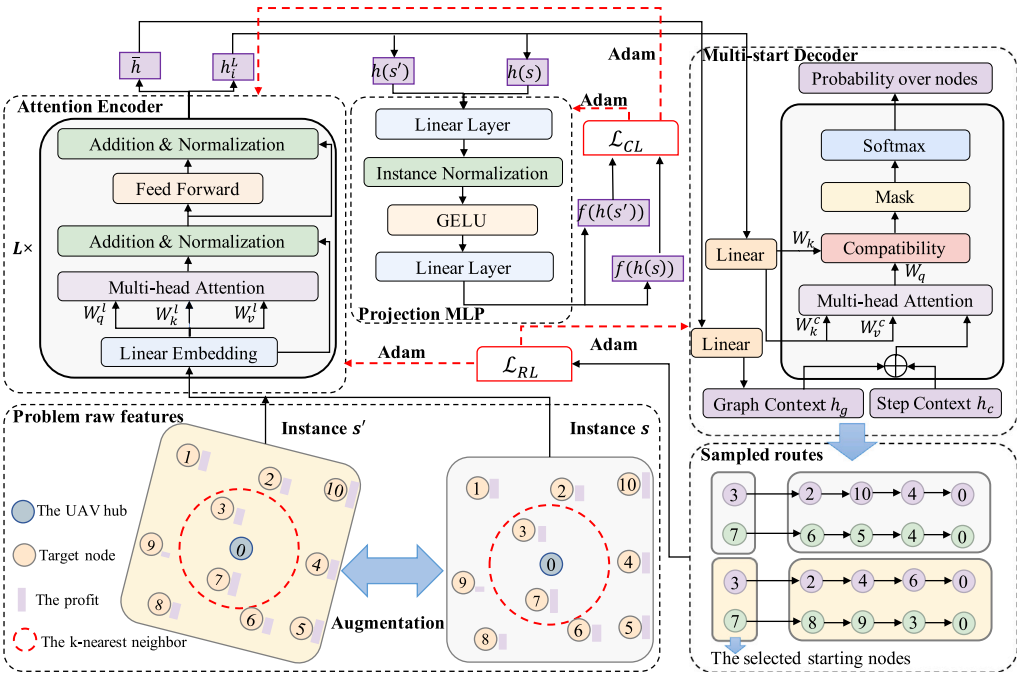


Fig. 1. The whole NNO-DRL structure consists of an attention encoder, a projection MLP, and a multi-start decoder.

4.1. NNO-DRL routing method for individual UAV

The routing solution usually consists of a sequence of nodes, which can be achieved by considering it as a sequential decision-making problem. In our method, we formulate the routing for each UAV as a Markov decision process with discrete time steps. Specifically, a UAV agent observes the environment state (*e.g.*, the target information and the remaining flight range) and then decides the next action to perform (*i.e.*, which target node to visit) according to its policy. Then the state is updated and the agent gathers the reward (profit) associated with the taken action. NNO-DRL aims to learn a policy p_θ for the agent to construct the route τ that maximizes the total reward (*i.e.*, the total profits) while respecting the constraints as described in Section 3. The probability of producing a route τ given an instance s can be expressed by the chain rule as follows,

$$p_\theta(\tau|s) = \prod_{t=1}^{T} p_\theta\left(\tau_t|s, \tau_{1:t-1}\right), \tag{12}$$

where $\tau = \tau_{1:T} = (\tau_1, \ldots, \tau_T)$ and T is the maximum number of states in the route construction process.

NNO-DRL consists of an adapted attention model, based on [18], and a projection MLP, as illustrated in Fig. 1. The adapted attention model, composed of an attention encoder and a multi-start decoder, is employed to estimate the policy $p_\theta(\tau|s)$. The key innovations in NNO-DRL include the integration of a projection MLP and CL for sophisticated problem representation and a multi-start decoder with a k-nearest neighbors strategy for improved solution space exploration. Specifically, the routing problem is framed as a graph with rotational invariance, where the original instance and its rotated augmentations share the same representation after processing through the attention encoder. To achieve this, we use a projection MLP and CL to maximize cosine similarity between node embeddings of the original instance and its augmentations, resulting in a more robust representation. Since generating multiple trajectories increases the likelihood of finding an optimal solution [52] and selecting distant nodes early in the route may lead to suboptimal solutions due to UAV battery constraints, we employ the multi-start decoder with a k-nearest neighbors strategy, which sets the first nodes as the k-nearest neighbors of the UAV hub, sampling k trajectories during both training and testing.

Given an instance s, NNO-DRL applies instance augmentation to obtain augmented instances. The attention encoder then extracts node and graph embeddings, which serve as inputs to the projection MLP and multi-start decoder. The multi-start decoder constructs routes step by step, while a joint training method combining CL and the REINFORCE algorithm is used to update the adapted attention model and projection MLP. The following sections detail the instance augmentation, attention encoder, projection MLP, multi-start decoder, and joint training algorithm.

4.1.1. Instance augmentation

Instance augmentation allows us to transform an instance s into multiple augmentations while reserving the same optimal solution, achieved by rotating or flipping all locations of target nodes in instance s. We exploit instance augmentation during both the training and inference stages. In training, we enhance the representation capability of the attention encoder by maximizing agreement among different augmentations of an instance [63]. During inference, we take advantage of the diverse solutions generated through various augmentations, effectively improving the probability of finding optimal solutions. Following [53,64], we rotate all node locations in instance s to obtain an augmentation s'. Specifically, we view all node locations as a graph, and rotate the graph at a random angle about its center as follows,

$$\begin{pmatrix} loc_x' \\ loc_y' \end{pmatrix} = \begin{pmatrix} (loc_x - o_x)\cdot\cos\theta - (loc_y - o_y)\cdot\sin\theta + o_x \\ (loc_x - o_x)\cdot\sin\theta + (loc_y - o_y)\cdot\cos\theta + o_y \end{pmatrix}, \theta \in [0, 2\pi) \tag{13}$$

where (loc_x, loc_y) and (o_x, o_y) are the node locations and the center of the graph, respectively, and (loc_x', loc_y') is the node locations of the augmented instance s'. In this paper, we generate a set of augmented instances $S = (s^1, \ldots, s^{\mathbb{M}})$ for a given instance s, where $\mathbb{M}$ is the number of augmentations and s^1 is the original view of instance s, *i.e.*, $s^1 = s$.

4.1.2. Attention encoder

For each UAV, the attention encoder takes the target information as the input, *i.e.*, 2-dim geographic location (loc_x^i, loc_y^i) and 1-dim profit p_i, and outputs node embeddings and a graph embedding. In particular, it first transforms the 3-dimensional information of targets (loc_x^i, loc_y^i, p_i), $i \in \{0, 1, \ldots, n\}$ into d_h-dim($d_h = 128$) initial embeddings h_i^0 through a

linear projection as follows,

$$h_i^0 = W_0\left(loc_x^i, loc_y^i, p_i\right) + b_0. \tag{14}$$

Then, the initial embeddings are advanced through L multi-head attention (MHA) layers to yield the advanced node embeddings $h_0^L, h_1^L, \ldots, h_n^L$. The lth MHA layer is structured by an MHA sublayer and a feed-forward (FF) sublayer, with their outputs processed by the skip-connection and batch normalization (BN) as follows,

$$\hat{h}_i^l = BN\left(h_i^{l-1} + MHA^l\left(h_i^{l-1}\right)\right), i \in \{1, \ldots, n\} \tag{15}$$

$$h_i^l = BN\left(\hat{h}_i^l + FF^l\left(\hat{h}_i^l\right)\right). \tag{16}$$

Finally, the encoder outputs node embeddings h_i^L and their average, *i.e.*, the graph embedding $\bar{h}$ as follows,

$$\bar{h} = mean\left(h_0^L, h_1^L, \ldots, h_n^L\right). \tag{17}$$

We set M_h(M_h=8) heads for all MHA sublayers such that each target will potentially receive messages from others. Regarding the lth MHA sublayer, we compute the *queries*, *keys*, and *values* via separate linear projections for each target as follows,

$$q_i^l = W_q^l h_i^{l-1}, k_i^l = W_k^l h_i^{l-1}, v_i^l = W_v^l h_i^{l-1}, \tag{18}$$

where W_q^l, W_k^l and W_v^l are trainable parameters in the lth layer. Then, we aggregate the messages received by target i with head $\gamma(\gamma \in \{1, \ldots, M_h\})$ as follows,

$$u_{i,j} = \frac{q_i^{l\top} k_j^l}{\sqrt{d_k}}, \tag{19}$$

$$a_{i,\gamma}^l = \sum_j \frac{e^{u_{i,j}}}{\sum_{j'} e^{u_{i,j'}}} v_j^l, \tag{20}$$

where $d_k = d_h/M_h = 16$ is the dimension of keys. The final attention value of node i is calculated as follows,

$$a_i^l = W_o Cat(a_{i,1}^l, \cdots, a_{i,M_h}^l), \tag{21}$$

where W_o is the trainable parameter and Cat means concatenation operator.

4.1.3. Projection MLP

The projection MLP (f) consists of two fully connected (FC) layers with instance normalization (IN) and GELU activation function [65] applied to the first layer. The input/output/hidden dimensions of the projection MLP are equal to that of the attention encoder (*i.e.*, 128). The projection MLP takes the node embeddings $h(s)$ and $h(s')$ (obtained from the attention encoder for instances s and s', respectively) as inputs and calculates their hidden representations $f(h(s))$ and $f(h(s'))$ using the following equation (Eq. (22)).

$$f(h(s)) = W_{p2}(\text{GELU}(\text{IN}(W_{p1} h(s)))) \tag{22}$$

where W_{p1} and W_{p2} are trainable parameters. The output of the projection MLP is used to calculate cosine similarity between the instance s and its augmented instances s' during the training process while projection MLP is excluded at the inference stage.

4.1.4. Multi-start decoder

Given the yielded node embeddings h_i^L and graph embedding $\bar{h}$ from the encoder, we propose a multi-start decoder that samples multiple trajectories simultaneously [52] and specify their first nodes as the k-nearest neighbors around the UAV hub, for better exploring near-optimal solutions.

Given the first targets for k trajectories, *i.e.*, $\tau^1, \tau^2, \ldots, \tau^k$, we construct the whole trajectories in parallel. Specifically, we utilize the current node embedding h_{t-1} and the remaining flight range L_t of the UAV to represent the context h_c at the current time step t. Then, h_c is added with the graph context h_g, *i.e.*, the linear transformation of graph embedding $\bar{h}$, to attain the query q_d for the decoding step as follows,

$$h_c = W_c\left[h_{t-1}; L_t\right], h_g = W_g \bar{h}, q_d = h_c + h_g, \tag{23}$$

ALGORITHM 1: Joint training method for NNO-DRL

Input: Training dataset $\mathbb{D}$, number of epochs $\mathbb{E}$, steps per epoch $\mathbb{T}$, batch size B.
Output: The trained model.

Initialize the parameter θ_{enc} of encoder, parameter θ_{mlp} of projection MLP, parameter θ_{dec} of decoder;
for *epoch = 1, 2, ⋯,* $\mathbb{E}$ **do**
  for *step = 1, 2, ⋯,* $\mathbb{T}$ **do**
    $s_b \leftarrow$ Sample from dataset ($\mathbb{D}$), $\forall b \in \{1, \cdots, B\}$;
    $\left\{s_b^1, s_b^2, \cdots, s_b^{\mathbb{M}}\right\} \leftarrow$ Augment(s_b), $\forall i \in \{1, \cdots, B\}$;
    $f(h(s_b^i)) \leftarrow$ Projection MLP(Encoder(s_b^i)), $\forall i \in \{1, \cdots, \mathbb{M}\}, \forall b \in \{1, \cdots, B\}$;
    $\left\{a_1^1, a_1^2, \cdots, a_1^k\right\}_b^i \leftarrow k$-nearest neighbor $\left(s_b^i\right)$, $\forall i \in \{1, \cdots, \mathbb{M}\}, \forall b \in \{1, \cdots, B\}$;
    $\tau_{i,b}^j \leftarrow$ Sample trajectories $\left(a_1^j, s_b^i, \theta_{enc}, \theta_{dec}\right)$, $\forall j \in \{1, \cdots, k\}, \forall i \in \{1, \cdots, \mathbb{M}\}, \forall b \in \{1, \cdots, B\}$;
    % Update θ_{enc} and θ_{mlp}
    $\mathcal{L}_{CL}^{\theta_{enc}, \theta_{mlp}} = -\frac{k_{CL}}{(\mathbb{M}-1)B} \sum_{b=1}^{B} \sum_{i=1}^{\mathbb{M}-1} \mathcal{D}(f(h(s_b^1)), f(h(s_b^{i+1})))$;
    $\theta_{enc} \leftarrow \text{Adam}(\theta_{enc}, \nabla \mathcal{L}_{CL}^{\theta_{enc}, \theta_{mlp}})$;
    $\theta_{mpl} \leftarrow \text{Adam}(\theta_{mpl}, \nabla \mathcal{L}_{CL}^{\theta_{enc}, \theta_{mlp}})$;
    % Update θ_{enc} and θ_{dec}
    $\nabla \mathcal{L}_{RL}^{\theta_{enc}, \theta_{dec}} = \frac{1}{k\mathbb{M}B} \sum_{b=1}^{B} \sum_{i=1}^{\mathbb{M}} \sum_{j=1}^{k} (R(\tau_{i,b}^j) - \frac{1}{k\mathbb{M}} \sum_{i=1}^{\mathbb{M}} \sum_{j=1}^{k} R(\tau_{i,b}^j)) \nabla_\theta \log p_\theta\left(\tau_{i,b}^j\right)$;
    $\theta_{enc} \leftarrow \text{Adam}(\theta_{enc}, \nabla \mathcal{L}_{RL}^{\theta_{enc}, \theta_{dec}})$;
    $\theta_{dec} \leftarrow \text{Adam}(\theta_{dec}, \nabla \mathcal{L}_{RL}^{\theta_{enc}, \theta_{dec}})$;
  end
end

where W_c and W_g refer to the trainable parameters, and [;] means the horizontal concatenation operator. Furthermore, the decoder projects the node embeddings (*i.e.*, $h_0^L, h_1^L, \ldots, h_n^L$) into keys K^c and values V^c by a liner transformation layer. Then, the query q_d, keys K^c and values V^c are processed by the MHA layer for message passing as follows,

$$f_\gamma^g = V^c softmax\left(\frac{K^c q_d}{\sqrt{d_k}}\right), \tag{24}$$

$$f^g = W_o^g Cat(f_1^g, f_2^g, \ldots, f_{M_h}^g), \tag{25}$$

where W_o^g is the trainable parameter. Finally, the output probability over targets is computed using single-head attention (M_h=1 so $d_k = d_h$) with a masked softmax, which masks the nodes (targets) that have been visited or cannot be visited within the remaining flight range,

$$q = W_q f^g, k_i = W_k h_i^L, \tag{26}$$

$$u_i^c = C \cdot tanh\left(\frac{q^\top k_i}{\sqrt{d_k}}\right), \tag{27}$$

$$p_i^t = \frac{e^{u_i^c}}{\sum_j e^{u_j^c}}, \tag{28}$$

where C is the clipping range ($C = 10$). The decoder will be applied multiple times until it reaches the final state in which the UAV comes back to the hub. We would like to note that during the training, we sample targets for trajectories in parallel following their policies, *i.e.*, probability distributions over targets. However, we greedily choose the targets with the highest probabilities for all trajectories during inference, so that the trained network only needs the feed-forward computation once to arrive at the next targets in all trajectories simultaneously.

4.1.5. Joint training method

The entire framework of the proposed NNO-DRL is detailed in the preceding sections. A joint training method is proposed to learn the model, consisting of a REINFORCE algorithm and a CL strategy. To make a clear introduction to them, we assume that the parameter related to the attention encoder, the multi-start decoder, and the projection MLP is denoted by θ_{enc}, θ_{dec} and θ_{mlp}, respectively. In the following, we introduce how to optimize these parameters.

Contrastive loss. In most learning-based methods, the same node embeddings extracted by the encoder are used to construct the whole solution. Consequently, the representation capability of the encoder plays a crucial role in obtaining high-quality solutions. CL, as a recent unsupervised learning method, has achieved significant success in computer vision and natural language processing [66–68] by extracting crucial features from unlabeled data for diverse tasks. In this paper, we leverage CL to enhance the quality of node embeddings extracted by the attention encoder. The attention encoder takes the augmentations $(s^1, \ldots, s^{\mathbb{M}})$ of instance s as the input and outputs their node embeddings $(h(s^1), \ldots, h(s^{\mathbb{M}}))$. Then, the projection MLP $f(\cdot)$ maps the node embeddings $(h(s^1), \ldots, h(s^{\mathbb{M}}))$ to the space where contrastive loss is applied, resulting the latent representations $(f(h(s^1)), \ldots, f(h(s^{\mathbb{M}})))$. Let cosine similarity between the instance and its augmentation evaluate their agreement as follows,

$$\mathcal{D}(f(h(s^1)), f(h(s^j))) = \frac{f(h(s^1))}{\|f(h(s^1))\|_2} \cdot \frac{f(h(s^j))}{\|f(h(s^j))\|_2}, j \in \{2, \ldots, \mathbb{M}\}, \quad (29)$$

where $\|\cdot\|$ is ℓ_2-norm. Then the contrastive loss function is defined as,

$$\mathcal{L}_{CL}^{\theta_{enc},\theta_{mlp}} = -\frac{k_{CL}}{(\mathbb{M}-1)} \sum_{i=1}^{\mathbb{M}-1} \mathcal{D}(f(h(s^1)), f(h(s^{i+1}))), \quad (30)$$

where $k_{CL}(k_{CL} = 0.1)$ is the coefficient of contrastive loss. Based on the contrastive loss $\mathcal{L}_{CL}^{\theta_{enc},\theta_{mlp}}$, the parameter θ_{enc} of the attention encoder and θ_{mlp} of the projection MLP are updated using Adam optimizer [69].

REINFORCE loss. NNO-DRL uses the adapted attention model to construct the solution. We exploit the REINFORCE algorithm, which is a classic policy gradient method, to train the attention model after a complete solution is constructed. The REINFORCE loss is defined as follows,

$$\mathcal{L}_{RL}^{\theta_{enc},\theta_{dec}} = -\mathbb{E}_{s' \sim S} \mathbb{E}_{\tau \sim p_\theta(\tau|s')}[R(\tau)], \quad (31)$$

where $R(\tau)$ denotes the total reward of the route τ. The REINFORCE loss $\mathcal{L}_{RL}^{\theta_{enc},\theta_{dec}}$ is optimized by the gradient descent, and the gradient for minimizing the REINFORCE loss is obtained as,

$$\nabla \mathcal{L}_{RL}^{\theta_{enc},\theta_{dec}} = -\mathbb{E}_{s' \sim S}[\mathbb{E}_{\tau \sim p_\theta(\tau|s')}[R(\tau) - \tilde{R}(s')] \nabla_\theta p_\theta(\tau|s')], \quad (32)$$

where $\tilde{R}(s')$ denotes the baseline for reducing the variance of the sampled gradients. We use Monte Carlo sampling to approximate $\nabla \mathcal{L}_{RL}^{\theta_{enc},\theta_{dec}}$. In specific, given an instance s with its augmentation set S and k sampled routes, we calculate the approximate gradient as below,

$$\nabla \mathcal{L}_{RL}^{\theta_{enc},\theta_{dec}} \approx -\frac{1}{k\mathbb{M}} \sum_{i=1}^{\mathbb{M}} \sum_{j=1}^{k} (R(\tau_i^j) - \tilde{R}(s)) \nabla_\theta \log p_\theta \left(\tau_i^j | s^i\right) \quad (33)$$

where $\tilde{R}(s) = \frac{1}{k\mathbb{M}} \sum_{i=1}^{\mathbb{M}} \sum_{j=1}^{k} R(\tau_i^j)$. Also, the parameter θ_{enc} of the attention encoder and θ_{dec} of the multi-start decoder are updated based on the REINFORCE loss $\mathcal{L}_{RL}^{\theta_{enc},\theta_{dec}}$ using Adam optimizer.

Algorithm 1 presents the pseudocode for the joint training method of NNO-DRL. The model parameters are initialized, and the model undergoes training for $\mathbb{E}$ epochs, each consisting of $\mathbb{T}$ steps. During each step, a batch of instances $\{s_1, \ldots, s_B\}$ is randomly sampled from the training dataset $\mathbb{D}$, and an augmentation set $\{s_b^1, s_b^2, \ldots, s_b^{\mathbb{M}}\}$ is generated for each instance $s_b \in \{s_1, \ldots, s_B\}$. Consequently, the model concurrently handles $\mathbb{M} \cdot B$ instances and generates k routes for each instance s_b^i following the policy $p_\theta(\tau_{i,b}^j | s_b^i)$. Finally, the model parameters are collaboratively updated using the CL strategy and REINFORCE algorithm.

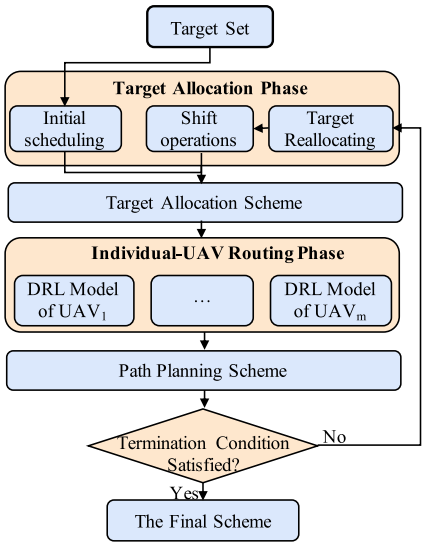


Fig. 2. The structure of the iterative two-phase framework.

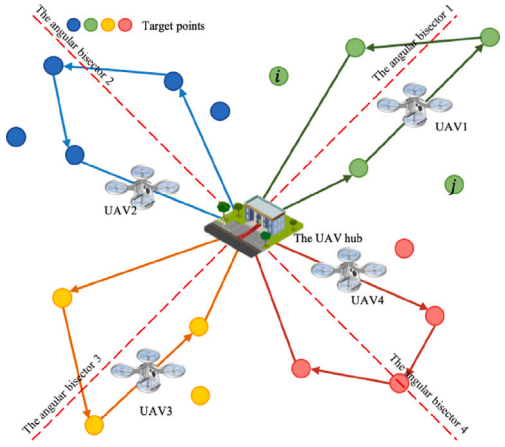


Fig. 3. The target allocation scheme in which the targets of the same color are allocated to the same UAV.

4.2. SA for target allocation

We outline the pseudocode of the SA-NNO-DRL in Algorithm 2. Given the UAV hub at the center of the area, we evenly divide all targets in the area into m groups (*i.e.*, m is also the number of UAVs) in terms of the angles between the target nodes and the horizontal axis starting from the hub(line 1), which is defined as

$$angle_i = arctan \frac{loc_y^i - loc_y^0}{loc_x^i - loc_y^0}, \quad (34)$$

where (loc_x^i, loc_y^i) refer to the coordinates of target i; (loc_x^0, loc_y^0) refer to the coordinates of the hub. Accordingly, each group of targets is initially assigned to the respective UAV(line 2).

After the targets are assigned to UAVs, the trained DRL model will perform the routing for all individual UAVs in parallel. Due to the maximum flight range restriction, a UAV might be unable to monitor all targets assigned to it. However, it is possible that others could monitor the uncovered targets through reallocation and rerouting. To this end, we exploit a SA algorithm to reallocate the target allocation scheme. The uncovered targets in the preceding routing phase would be fed back to the target allocation phase for reassignment(line 7). With specifically designed shift operations between different UAVs, the target allocation is adjusted and the DRL model also accordingly amends the routing

ALGORITHM 2: SA-NNO-DRL

Input: Initial temperature T_s, the lowest temperature T_e, Markov chain length $\mathbb{L}$, decreasing rate σ.
Output: The final scheme X_{best}.

1 Generate an initial target allocation scheme $X(\chi_1, \cdots, \chi_m)$;
2 Record uncovered target set $\mathbb{T}(\xi_1, \cdots, \xi_m)$, and obtain the value of objective function $fit \leftarrow \{DRL\_model(\chi_1), \cdots, DRL\_model(\chi_m)\}$;
3 $X_{best} = X, fit_{best} = fit$;
4 Initialize $l = 1, T_o = T_s$;
5 **while** $T_o > T_e$ **do**
6 **for** $l = 1 : \mathbb{L}$ **do**
7 Generate a new target allocation scheme X' by shifting the uncovered targets ξ_i between UAVs, where $i \in \{1, 2, \cdots, m\}$;
8 Record uncovered target set $\mathbb{T}'(\xi_1', \cdots, \xi_m')$, and obtain the value of objective function $fit' \leftarrow \{DRL\_model(\chi_1'), \cdots, DRL\_model(\chi_m')\}$;
9 $\Delta f = fit' - fit$;
10 **if** $\Delta f > 0$ **then**
11 $X = X', fit = fit'$;
12 **if** $fit > fit_{best}$ **then**
13 $X_{best} = X, fit_{best} = fit$;
14 **end**
15 **end**
16 **else if** $e^{\frac{\Delta f}{T_o}} > \epsilon, \epsilon \leftarrow uniform(0, 1)$ **then**
17 $X = X'$;
18 **end**
19 **end**
20 $T_o = \sigma T_o$
21 **end**

for all UAVs(line 8). The reallocation scheme is evaluated using the SA criterion: if the new allocation yields higher profits, it is accepted(lines 10–15); otherwise, it may still be accepted with a small probability which is determined by the difference in profits between the new and old schemes and the current annealing temperature(lines 16–18). The above reallocation and rerouting are iterated until the terminal condition is satisfied as displayed in Fig. 2, and details of the shift operation are described below.

Shift operation. In the initial target allocation, we define the angle of each target node as $angle_i$ and determine the angular bisector of the subarea for each UAV. Based on the angles and angular bisectors, we exploit a shift operator, which passes partially uncovered targets of a UAV to its neighbor UAV for rerouting in the next iteration. With the uncovered targets in the left and right part of the angular bisector, we define the shift operator with an adjustable pass direction as follows, (1) the pass direction is clockwise at even iterations and the uncovered targets in the right part for each UAV are passed to its right neighbor; (2) the pass direction is counterclockwise at odd iterations and the uncovered targets in left part for each UAV are passed to its left neighbor. An illustration of the shift operation with four UAVs is depicted in Fig. 3, where UAV1 cannot cover target i and target j in the current iteration. If the number of the next iteration is even, target j will be passed to UAV4, and target i will remain unassigned; otherwise, target i will be passed to UAV2, and target j will remain unassigned.

Since more than one UAV might be routed in each routing phase, and each UAV might be assigned with different numbers of targets, we append dummy UAV hubs (in terms of the features such as the location and profit) to the end of corresponding input sequences to make them have the same length for better parallel computation. Note that the extra dummy UAV hub is masked so as not to be selected during the k-nearest neighbors selection and decoding process.

Table 1
Hyperparameters in SA-NNO-DRL algorithm.

Parameters	Phase	Value
The number of epochs $\mathbb{E}$	DRL	10
The size of training dataset $\lvert\mathbb{D}\rvert$	DRL	1280000
The size of validation dataset $\lvert\mathbb{D}_v\rvert$	DRL	10 000
The batch size B	DRL	64
The number of encoder layers L	DRL	6
The decaying rate δ	DRL	0.995
The Markov chain length $\mathbb{L}$	SA	4
The initial temperature T_s	SA	80
The lowest temperature T_e	SA	60
The temperature decreasing rate σ	SA	0.9

5. Experiments

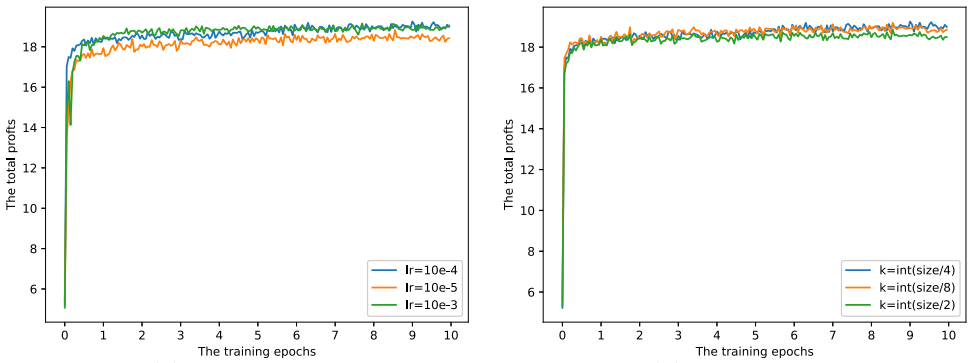
Our experimental evaluation consists of multiple steps. Firstly, we evaluate the performance of our SA-NNO-DRL approach in solving MURMPPs, considering varying sizes and profit types of monitoring targets. Subsequently, we demonstrate the generalization ability of SA-NNO-DRL by applying it to MURMPP instances with different numbers of targets and UAVs, as well as varying hub locations. Additionally, we conduct an ablation study to analyze the contribution of each component in the SA-NNO-DRL approach. Lastly, we evaluate the effectiveness of the proposed NNO-DRL routing method for individual UAVs by comparing it against several state-of-the-art routing methods.

5.1. Experimental settings

We describe the data generation and hyperparameters for training the SA-NNO-DRL model and then demonstrate the training curves.

Data generation. For MURMPP instances, the coordinates of targets are generated by uniform sampling from a unit square $[0, 1]^2$ following the conventions [18,52] and the UAV hub is located at the center (*i.e.*, $[0.5, 0.5]$). However, our method can handle instances with different locations of UAV hubs well, which we will verify in the subsequent experiments. We assume that all UAVs are homogeneous, and their maximum flight range is set to 2. To comprehensively evaluate the performance of our SA-NNO-DRL, we consider two types of profits for MURMPPs, (1) *constant* profits with each target bearing the same value 1; (2) *uniform* profits with each target bearing a value uniformly sampled from $[0, 1]$. In both cases, the profit of the UAV hub is set to 0. While MURMPPs with constant profits tend to monitor as many targets as possible, the uniform ones encourage the UAV to focus more on targets with higher profits. For each type of profits, we assess our SA-NNO-DRL on different problem sizes, *i.e.*, 40 targets with 2 UAVs, 60 targets with 3 UAVs, 80 targets with 4 UAVs, 100 targets with 5 UAVs, 150 targets with 5 UAVs, and 200 targets with 5 UAVs, and term them as T40U2, T60U3, T80U4, T100U5, T150U5, and T200U5, respectively. Taking T40U2 with constant and uniform profits as an example, we further term them as T40U2C and T40U2U, respectively, for simplification.

Hyperparameters. The hyperparameters of SA-NNO-DRL are shared to train DRL policies for all problem sizes. Similar to [18], the training instances are generated on the fly in each epoch. Since the NNO-DRL model is trained to route for each UAV, we first generate multiple MURMPP instances and allocate targets to UAVs in each instance by the target allocation method introduced in Section 4.2. The allocated targets for each UAV in all instances are gathered as a training dataset. Since the target allocation method assigns each UAV the same number of targets, the number of targets are 20, 20, 20, 20, 30, and 40 for T40U2, T60U3, T80U4, T100U5, T150U5, and T200U5, respectively. We optimize the NNO-DRL model with Adam [69]. During the training and testing, the number of augmentations for each instance (*i.e.*, $\mathbb{M}$) is set to 4. We set the initial learning rate lr to 10^{-4} and define the number of trajectories with specified start nodes as a quarter of



(a) Different lr (b) Different k

Fig. 4. Training curves with different hyperparameters.

Table 2
Results on MURMPP instances with different problem sizes and profit types.

Instance group		Gurobi			OR-Tools			GAMMA(Greedy)			GAMMA(1280)			AM(Greedy)			AM(1280)			SA-NNO-DRL		
		Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Constant profits	T40U2C	**30.87**	**0.00%**	1083 s	28.67	7.13%	0.05 s	28.53	7.56%	0.02 s	29.80	3.46%	1.56 s	28.90	6.37%	0.03 s	29.27	5.17%	0.33 s	30.37	1.62%	0.22 s
	T60U3C	51.30	2.47%	1800 s	50.93	3.17%	0.16 s	50.03	4.88%	0.03 s	51.87	1.39%	0.65 s	47.73	9.25%	0.03 s	49.20	6.46%	0.48 s	**52.60**	**0.00%**	0.25 s
	T80U4C	73.77	2.73%	1800 s	73.40	3.21%	0.34 s	73.07	3.65%	0.04 s	75.20	0.84%	0.76 s	67.93	10.42%	0.04 s	69.97	7.74%	0.67 s	**75.83**	**0.00%**	0.25 s
	T100U5C	94.50	3.90%	1800 s	**98.33**	**0.00%**	0.65 s	94.57	3.83%	0.06 s	97.23	1.12%	1.49 s	89.27	9.22%	0.05 s	92.50	5.93%	0.41 s	98.07	0.27%	0.24 s
	T150U5C	115.10	16.17%	1800 s	131.20	4.44%	1.88 s	117.83	14.18%	0.09 s	128.57	6.36%	1.33 s	105.47	23.19%	0.08 s	114.73	16.44%	0.51 s	**137.30**	**0.00%**	0.34 s
	T200U5C	128.73	23.90%	1800 s	158.93	6.05%	3.60 s	144.53	14.56%	0.17 s	154.47	8.69%	1.83 s	128.67	23.94%	0.10 s	137.43	18.76%	0.69 s	**169.17**	**0.00%**	0.44 s
Uniform profits	T40U2U	**16.53**	**0.00%**	1337 s	15.60	5.59%	0.05 s	15.44	6.54%	0.02 s	16.05	2.90%	0.77 s	15.05	8.93%	0.02 s	15.58	5.70%	0.18 s	15.95	3.50%	0.22 s
	T60U3U	**28.32**	**0.00%**	1800 s	27.83	1.72%	0.15 s	26.79	5.41%	0.04 s	27.81	1.78%	0.91 s	25.69	9.30%	0.03 s	26.44	6.62%	0.34 s	28.13	0.66%	0.25 s
	T80U4U	38.40	1.86%	1800 s	38.65	1.21%	0.33 s	37.96	2.97%	0.04 s	38.67	1.16%	1.16 s	35.54	9.17%	0.04 s	36.49	6.73%	0.45 s	**39.13**	**0.00%**	0.24 s
	T100U5U	50.02	0.95%	1800 s	50.23	0.53%	0.56 s	48.74	3.49%	0.07 s	50.00	0.97%	1.35 s	46.67	7.58%	0.05 s	48.07	4.81%	0.61 s	**50.50**	**0.00%**	0.25 s
	T150U5U	60.57	15.29%	1800 s	70.06	2.02%	2.49 s	62.56	12.50%	0.07 s	67.27	5.92%	1.71 s	58.80	17.76%	0.08 s	63.53	11.15%	0.50 s	**71.50**	**0.00%**	0.33 s
	T200U5U	70.79	21.88%	1800 s	86.70	4.32%	4.38 s	77.70	14.26%	0.10 s	83.21	8.18%	2.18 s	72.06	20.49%	0.10 s	77.63	14.33%	0.56 s	**90.62**	**0.00%**	0.44 s

The **bold** means the best objective value and gap in each row, throughout the paper.

the problem size (*i.e.*, $k = \text{int}(size/4)$). The learning curves of our method with different lr and k values on T100U5C are shown in Fig. 4(a) and 4(b), respectively. Results indicate that $lr = 10^{-4}$ and $k = \text{int}(size/4)$ achieve the best performance. For clarity, other SA-NNO-DRL hyperparameters are listed in Table 1. We would like to note that the runtime of our methods is primarily affected by two key factors: the number of iterations of the SA and the number of augmentations used. Increasing either of these factors generally leads to a longer runtime, requiring a careful balance between computational time and potential performance gains. Our method is implemented with Pytorch and runs on a workstation with an i9-9900K CPU and a single GTX3090 GPU. Our implementation code is publicly available[1]

Training curves. At first glance, we plot the training curves in terms of rewards (*i.e.*, the total profits collected by a UAV) for our DRL model in all problem sizes. Training curves on instances with constant profits and uniform profits are shown in Fig. 5 and Fig. 6, respectively. In general, the NNO-DRL model increases the total profits stably along epochs for different problem sizes and profit types. While the number of targets in training datasets of T40U2, T60U3, T80U4, and T100U5 are all 20, the total profits collected by a UAV increase with the problem scale. The reason might be that the density of targets across the same area increases as the problem scale grows, so a UAV restricted by the same maximum flight range could visit more targets. Regarding T100U5, T150U5, and T200U5, the rewards increase with the problem scale mainly because the number of targets assigned to each UAV increases. Overall, the training curves indicate that our experimental settings work fairly well. In the following subsection, we use the trained models to solve respective instances with different problem sizes for comparison with baselines.

5.2. Comparison analysis

To evaluate the effectiveness of the proposed SA-NNO-DRL method, we adopt various baselines as follows:

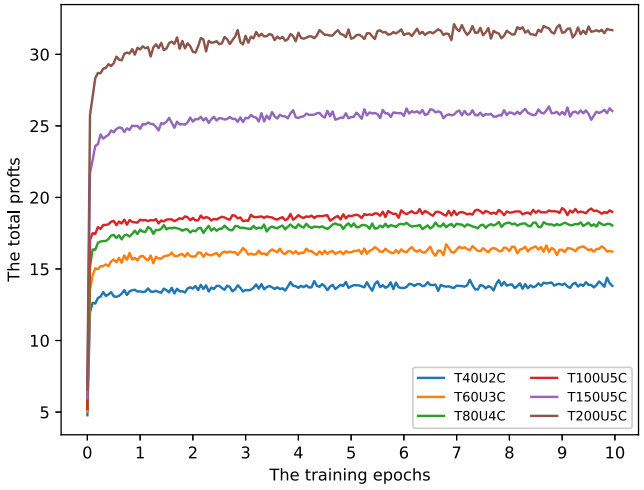


Fig. 5. Training curves with constant profits.

1. *Gurobi:* The well known commercial solver Gurobi [70] is used to solve the MILP formulated in Section 3. We set the maximum runtime of the solver to 30 min for each instance, to avoid a prohibitively long time for computing all instances in the whole testing set.
2. *OR-Tools:* The state-of-the-art heuristic based solver, OR-Tools [71], is widely applied for various routing problems. In the implementation for solving MURMPP, we replace the capacity constraint in OR-Tools with the constraint of the maximum flight range.
3. *GAMMA:* The state-of-the-art learning-based method [30] for solving multi-agent reconnaissance mission planning problems with multiple depots. We adapt GAMMA for solving MURMPP by setting the depot of each agent as the same one. We train

[1] https://github.com/mingfan321/SA-NNO-DRL

Table 3
The number of solutions found by each solver as the best solution.

Instance group		Gurobi	OR-Tools	GAMMA(Greedy)	GAMMA(1280)	AM(Greedy)	AM(1280)	SA-NNO-DRL
Constant profits	T40U2C	**28**	3	3	10	4	5	16
	T60U3C	10	3	1	11	0	1	**19**
	T80U4C	9	7	1	8	0	0	**17**
	T100U5C	1	**20**	0	10	0	0	15
	T150U5C	0	1	0	0	0	0	**30**
	T200U5C	0	2	0	0	0	0	**28**
Uniform profits	T40U2U	**24**	1	1	1	0	0	4
	T60U3U	**15**	7	0	1	0	0	8
	T80U4U	5	7	0	4	0	0	**14**
	T100U5U	5	9	0	2	0	0	**14**
	T150U5U	0	6	0	0	0	0	**24**
	T200U5U	0	1	0	0	0	0	**29**

The **bold** means the most number of best solutions in each row.

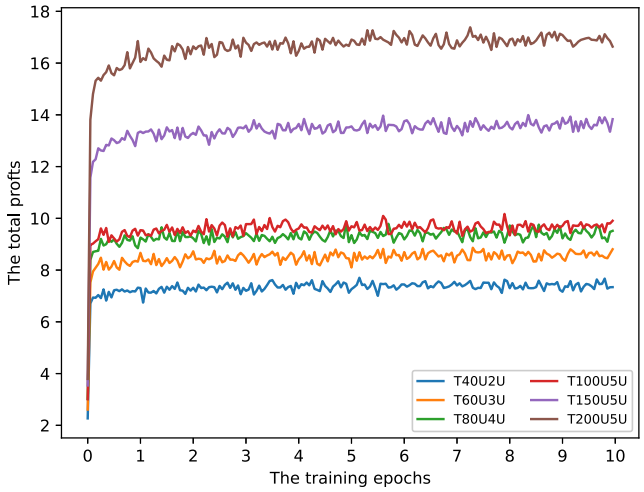


Fig. 6. Training curves with uniform profits.

GAMMA models for all problem sizes with original hyperparameters in [30] except for the batch size, which we set to 512 on T40U2C/U, T60U3C/U, T80U4C/U, T100U5C/U and 256 on T150U5C/U and T200U5C/U given the memory limit.

4. *AM:* We use AM model in [18] as another learning-based baseline. The original AM model can only solve a single-agent routing problem. Hence, for each MURMPP instance, we iteratively route each UAV by the AM and remove the selected targets, until all UAVs have been routed. The profits collected by all UAVs are the objective value of the original MURMPP. The AM model is trained on 20, 50, and 100 targets with two profit types (*i.e.*, constant and uniform profit), respectively.

All baselines are implemented in Python and executed on the same workstation as SA-NNO-DRL. Regarding the pure learning-based methods, *i.e.*, GAMMA and AM, we apply two types of decoding strategies during testing: (1) *Greedy*, in which the UAV always chooses the action (*i.e.*, the next target to visit) with the maximum probability at each decoding step; (2) *Sampling*, which engenders $\mathbb{N}$ solutions by sampling targets at each step according to the probability distribution, and then returns the one with the best objective value. In our experiments, we set $\mathbb{N}$ to 1280, and term GAMMA and AM with the 1280 samples as GAMMA(1280) and AM(1280), respectively. Since the exact solver (*i.e.*, Gurobi) consumes a very long time to solve a MURMPP instance, we only generate 30 instances with each profit type for each problem size in testing. These instances are shared by our SA-NNO-DRL method and the other baselines. We record the performance of our method and baselines on MURMPP instances with different sizes and profit types in Table 2, where the average objective value (Obj.), gap, and computation time are computed across the testing instances. The gap for a single instance is calculated by comparing the objective value of a method with the best one found among all methods. Furthermore, we present the number of solutions found by each solver as the best solution in Table 3. Notably, we use AM model trained for 20 targets (nodes) to test instances of T40U2, T60U3, T80U4, and T100U5. In addition, AM model trained for 50 targets (nodes) is used to test instances of T150U5 and T200U5.

Analysis of results with constant profits. For MURMPP instances with constant profits, Gurobi delivers the most number of best solutions on the smallest problem size, *i.e.*, T40U2C. However, given the predefined computation time, its performance degenerates as the problem size grows. Note that most instances in T40U2C are solved optimally by Gurobi within the predefined runtime so that the average time is less than 1800s in Table 2. Comparing the learning-based methods GAMMA and AM, we observe that the Sampling decoding strategy achieves larger objective values and smaller gaps than the Greedy one, with a slightly longer computation time. GAMMA(1280) attains significantly better performance than AM(1280) on all problem sizes. On the other hand, the classic heuristic solver OR-Tools outstrips AM(Greedy), AM(1280), and GAMMA(Greedy) on all problem sizes (except for T40U2C). It also outperforms GAMMA(1280) on large problem sizes (*i.e.*, T100U5C, T150U5C, and T200U5C). It indicates that learning-based methods and OR-Tools might have the potential to complement each other across different problem sizes. Our hybrid method, *i.e.*, SA-NNO-DRL, verifies this point by surpassing ORTools and all learning-based methods on T40U2C, T60U3C, T80U4C, T150U5C, and T200U5C. Meanwhile, the gap of SA-NNO-DRL is only 0.27% in the T100U5C, with the objective values very close to that of the best method, *i.e.*, 98.07 (SA-NNO-DRL) vs. 98.33 (OR-Tools). As for running efficiency, SA-NNO-DRL can solve problems of all sizes within 0.5 s. In summary, our SA-NNO-DRL method, which mixes heuristic with DRL, generally achieves the best results on MURMPP with constant profits with a very short computation time.

Analysis of results with uniform profits. As shown in the lower part of Table 2 and Table 3, the results for MURMPP with uniform profits share a similar pattern to the results with constant profits. Gurobi performs well on small problem instances (*i.e.*, T40U2U and T60U3U) in terms of objective values, but it has a long computation time. In contrast, our SA-NNO-DRL outperforms Gurobi on larger instances (T80U4U to T200U5U) in both solution quality and computation efficiency. Additionally, SA-NNO-DRL surpasses OR-Tools, GAMMA(Greedy), GAMMA(1280), AM(Greedy) and AM(1280) across all instance groups except T40U2U, where it falls short of GAMMA (1280).

Considering the results presented in Table 2 and Table 3, our comprehensive analysis reveals that the hybrid SA-NNO-DRL method exhibits the best overall performance compared to other methods, with

Table 4
Generalization on large problem sizes.

Instance group		Gurobi			OR-Tools			GAMMA(Greedy)			GAMMA(1280)			AM(Greedy)			AM(1280)			SA-NNO-DRL		
		Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time			
Constant profits	T300U5C	134.93	38.53%	1800 s	199.70	9.02%	5.64 s	180.97	17.56%	0.15 s	190.30	13.30%	13.23 s	140.00	36.22%	0.15 s	152.13	30.69%	9.80 s	**219.50**	**0.00%**	0.79 s
	T400U5C	121.53	53.33%	1800 s	238.00	8.60%	9.35 s	205.03	21.26%	0.18 s	208.90	19.78%	14.46 s	170.07	34.69%	0.20 s	178.83	31.32%	11.85 s	**260.40**	**0.00%**	1.08 s
	T500U5C	98.83	66.26%	1800 s	269.43	8.01%	14.39 s	222.73	23.96%	0.22 s	212.30	27.52%	16.20 s	196.33	32.97%	0.27 s	202.77	30.77%	13.86 s	**292.90**	**0.00%**	1.50 s
	T600U5C	93.93	70.63%	1800 s	296.47	7.30%	20.77 s	241.73	24.41%	0.27 s	209.97	34.34%	17.57 s	220.37	31.09%	0.34 s	218.60	31.64%	15.64 s	**319.80**	**0.00%**	1.97 s
	T700U5C	79.50	77.04%	1800 s	321.63	7.11%	28.56 s	254.23	26.57%	0.32 s	197.50	42.96%	17.88 s	241.03	30.38%	0.42 s	231.07	33.26%	18.04 s	**346.23**	**0.00%**	2.63 s
	T800U5C	–	–	–	348.30	5.86%	37.21 s	267.57	27.68%	0.37 s	194.43	47.45%	18.56 s	261.53	29.31%	0.52 s	240.50	34.99%	20.16 s	**369.97**	**0.00%**	3.39 s
	T900U5C	–	–	–	369.43	4.91%	49.48 s	279.17	28.14%	0.41 s	194.13	50.03%	19.71 s	283.57	27.01%	0.62 s	248.53	36.03%	22.52 s	**388.50**	**0.00%**	4.46 s
	T1000U5C	–	–	–	391.07	3.73%	56.28 s	290.43	28.50%	0.47 s	195.87	51.78%	19.23 s	298.83	26.43%	0.73 s	251.10	38.18%	24.90 s	**406.20**	**0.00%**	6.04 s
Uniform profits	T300U5U	75.03	37.83%	1800 s	110.22	8.67%	5.35 s	99.05	17.93%	0.18 s	106.77	11.53%	12.72 s	85.31	29.31%	0.15 s	93.13	22.83%	9.87 s	**120.68**	**0.00%**	0.83 s
	T400U5U	86.22	40.18%	1800 s	129.08	10.43%	9.60 s	117.23	18.65%	0.17 s	123.94	14.00%	14.73 s	101.81	29.36%	0.20 s	107.31	25.54%	11.18 s	**144.11**	**0.00%**	1.09 s
	T500U5U	46.29	71.71%	1800 s	145.77	10.90%	12.68 s	132.95	18.74%	0.22 s	135.94	16.91%	16.28 s	115.27	29.54%	0.30 s	118.75	27.42%	12.66 s	**163.61**	**0.00%**	1.43 s
	T600U5U	54.63	69.32%	1800 s	159.59	10.38%	17.42 s	143.15	19.61%	0.25 s	144.88	18.65%	17.76 s	126.21	29.13%	0.34 s	126.70	28.85%	14.45 s	**178.08**	**0.00%**	1.91 s
	T700U5U	51.45	73.36%	1800 s	172.44	10.72%	27.16 s	156.89	18.78%	0.30 s	154.49	20.01%	18.58 s	137.19	28.97%	0.46 s	134.28	30.48%	20.04 s	**193.15**	**0.00%**	2.33 s
	T800U5U	56.90	72.52%	1800 s	183.55	11.37%	29.56 s	165.85	19.92%	0.34 s	162.27	21.65%	20.10 s	144.46	30.25%	0.57 s	139.41	32.68%	18.28 s	**207.09**	**0.00%**	3.33 s
	T900U5U	38.77	82.17%	1800 s	194.35	10.64%	37.04 s	176.83	18.69%	0.39 s	169.42	22.11%	21.42 s	152.40	29.93%	0.63 s	143.86	33.85%	20.55 s	**217.49**	**0.00%**	4.41 s
	T1000U5U	–	–	–	204.16	9.50%	43.90 s	181.84	19.39%	0.44 s	174.05	22.85%	22.55 s	159.10	29.47%	0.75 s	147.75	34.51%	22.98 s	**225.59**	**0.00%**	5.38s

Table 5
Generalization on varying number sof UAVs.

Instance group		Gurobi			OR-Tools			GAMMA(Greedy)			GAMMA(1280)			AM(Greedy)			AM(1280)			SA-NNO-DRL		
		Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Constant profits	T200U2C	74.77	4.31%	1800 s	69.87	10.58%	0.71 s	68.70	12.07%	0.10 s	76.10	2.60%	0.86 s	62.90	19.50%	0.05 s	68.43	12.41%	1.66 s	**78.13**	**0.00%**	1.05 s
	T200U3C	101.77	10.78%	1800 s	101.00	11.46%	1.11 s	96.73	15.20%	0.07 s	106.77	6.40%	1.01 s	87.50	23.29%	0.07 s	95.37	16.39%	2.20 s	**114.07**	**0.00%**	0.71 s
	T200U4C	125.5	13.90%	1800 s	131.83	9.56%	1.84 s	122.53	15.94%	0.09 s	132.67	8.99%	1.19 s	109.63	24.79%	0.09 s	118.03	19.03%	2.36 s	**145.77**	**0.00%**	0.53 s
	T200U6C	130.90	29.96%	1800 s	181.00	3.16%	2.81 s	161.93	13.36%	0.11 s	171.87	8.04%	1.44 s	142.73	23.63%	0.11 s	154.50	17.34%	2.63 s	**186.90**	**0.00%**	0.36 s
	T200U7C	129.83	34.18%	1800 s	196.67	0.30%	3.18 s	174.10	11.74%	0.12 s	184.40	6.52%	1.58 s	155.30	21.27%	0.12 s	168.17	14.75%	3.69 s	**197.27**	**0.00%**	0.31 s
	T200U8C	124.87	37.57%	1800 s	**200.00**	**0.00%**	2.81 s	183.63	8.18%	0.13 s	192.30	3.85%	1.68 s	165.73	17.13%	0.13 s	178.67	10.67%	4.01 s	199.57	0.22%	0.27 s
Uniform profits	T200U2U	42.72	4.54%	1800 s	39.03	12.78%	0.50 s	37.58	16.03%	0.05 s	43.09	3.71%	0.82 s	36.04	19.45%	0.05 s	39.14	12.52%	1.27 s	**44.75**	**0.00%**	1.13 s
	T200U3U	57.83	9.03%	1800 s	57.04	10.27%	1.09 s	53.51	15.83%	0.07 s	59.08	7.05%	0.94 s	50.33	20.82%	0.07 s	54.26	14.65%	1.98 s	**63.57**	**0.00%**	0.72 s
	T200U4U	69.37	13.41%	1800 s	73.25	8.57%	1.72 s	66.12	17.47%	0.08 s	72.62	9.36%	1.09 s	62.05	22.55%	0.08 s	67.13	16.21%	2.46 s	**80.12**	**0.00%**	0.53 s
	T200U6U	71.73	26.51%	1800 s	95.83	1.82%	3.32 s	84.61	13.31%	0.10 s	90.70	7.07%	1.41 s	79.89	18.15%	0.11 s	86.05	11.84%	3.49 s	**97.60**	**0.00%**	0.38 s
	T200U7U	72.22	28.09%	1800 s	100.24	0.20%	3.25 s	90.74	9.65%	0.11 s	95.48	4.94%	1.40 s	86.03	14.34%	0.12 s	91.79	8.61%	3.89 s	**100.44**	**0.00%**	0.33 s
	T200U8U	72.86	27.91%	1800 s	100.74	0.31%	2.54 s	94.15	6.84%	0.12 s	98.13	2.90%	2.39 s	91.21	9.75%	0.13 s	96.20	4.81%	4.39 s	**101.06**	**0.00%**	0.29 s

the smallest gaps in 8 out of 12 instance groups. Additionally, SA-NNO-DRL demonstrates efficient computation, with a runtime of fewer than 0.5 s, while consistently delivering high-quality solutions.

5.3. Generalization analysis

The trained deep learning model is expected to generalize to different problem instances, which means it can be applied to various scenarios. To comprehensively verify the generalization performance of the proposed SA-NNO-DRL method, we conduct three types of experiments: (1) we evaluate the model learned for a fixed number of targets to solve instances with more targets; (2) we evaluate the model learned for a fixed number of UAVs to solve instances with different numbers of UAVs (*i.e.*, with smaller and larger fleet sizes); (3) we evaluate the model learned for a fixed location of UAV hub, to solve instances with different hub locations.

5.3.1. Generalization on larger problem sizes

We generate 30 instances with different numbers of targets and keep the number of UAVs the same, *i.e.*, T300U5, T400U5, T500U5, T600U5, T700U5, T800U5, T900U5, and T100U5, respectively. We also consider the constant and uniform profits for each problem size. In our comparison with all baselines, we exclude the results by Gurobi on T800U5C, T900U5C, T100U5C, and T1000U5U, as Gurobi fails to produce feasible solutions within the predefined runtime for these instances. We apply SA-NNO-DRL and GAMMA models learned for T200U5C/U to MURMPP instances with larger problem sizes. For a fair comparison, we apply the AM model learned for 100 targets (nodes) on larger problem sizes.

The experiment results are displayed in Table 4. Regarding the constant profits, our SA-NNO-DRL method consistently outperforms all other solvers across all group instances. Notably, OR-Tools achieves the second-best solutions but requires significantly longer computation time compared to SA-NNO-DRL. While GAMMA (Greedy) and AM (Greedy) demonstrate efficient solution computation, we observe that their performance gaps widen as the problem size increases, indicating their significant inferiority to SA-NNO-DRL and OR-Tools. On the other hand, we observe that the advantage of the Sampling decoding strategy for AM and GAMMA disappears as the problem scales up, which indicates that sampling 1280 solutions cannot find a better solution than Greedy given a prohibitive computation complex.

Regarding the uniform profits, SA-NNO-DRL also delivers the best results in all instance groups. One key observation from Table 4 is that the superiority of SA-NNO-DRL over OR-Tools is more pronounced in instances with uniform profits compared to those with constant profits. Additionally, the performance gaps between the pure learning-based methods (GAMMA and AM) and OR-Tools are relatively smaller in instances with uniform profits compared to instances with constant profits. In other words, all DRL-based methods perform better on MURMPP with uniform profits. It indicates that the deep neural network could effectively extract the features of the uniform profits, and the DRL agent could learn policies to choose targets wisely for maximizing the total profit.

Overall, the SA-NNO-DRL can efficiently attain desirable solutions in all large-scale problems even with 1,000 targets and reveals a good trade-off between solution quality and computation efficiency, which suggests a fairly favorable generalization. Notably, the NNO-DRL model in the hybrid SA-NNO-DRL method is only trained with 40 targets (nodes). Therefore, our method might deliver better solutions given the NNO-DRL model trained with a larger problem size. Our method also has the potential to solve instances with more than 1,000 targets.

5.3.2. Generalization to different team sizes of UAVs

We further evaluate SA-NNO-DRL and all baselines on MURMPP instances with different team sizes of UAVs. We generate 30 instances with different numbers of UAVs, *i.e.*, 200 targets with 2 UAVs, 3 UAVs, 4 UAVs, 6 UAVs, 7 UAVs, and 8 UAVs, termed as T200U2, T200U3, T200U4, T200U6, T200U7 and T200U8, respectively. We still consider constant and uniform profits under different numbers of UAVs. We evaluate SA-NNO-DRL and GAMMA models learned for T200U5C/U on each instance group. Meanwhile, the AM models learned for 50 targets under two profit types are evaluated on the same instance groups. The results are summarized in Table 5.

In all instance groups except for T200U8C, SA-NNO-DRL achieves the highest objective values, indicating its favorable generalization ability across different numbers of UAVs. While OR-Tools provides the best solutions for T200U8C, it is worth noting that the gap between SA-NNO-DRL and OR-Tools in this particular instance is only 0.22%. In

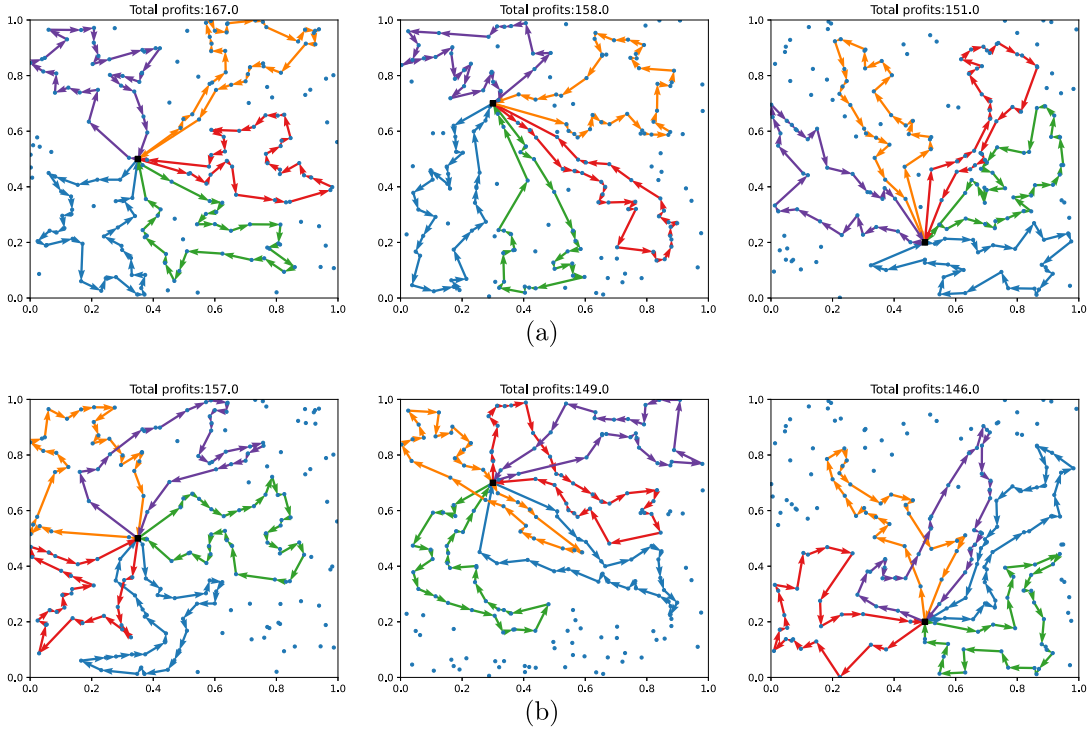


Fig. 7. The visualization of solutions by (a) SA-NNO-DRL and (b) OR-Tools for instances with different locations of UAV hubs.

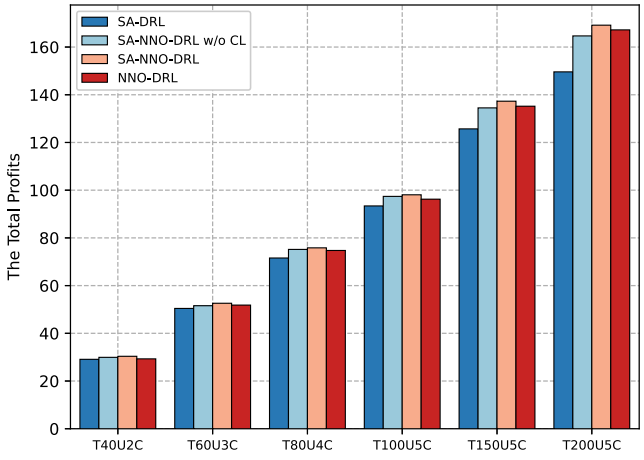


Fig. 8. Ablation study on hybrid models with constant profits.

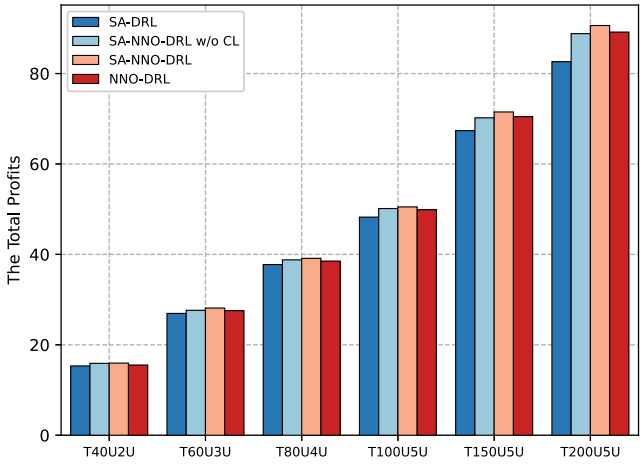


Fig. 9. Ablation study on hybrid models with uniform profits.

terms of computational efficiency, SA-NNO-DRL is much more efficient than Gurobi and surpasses GAMMA(1280), AM(1280), and OR-Tools on instances with a higher number of UAVs, such as T200U3C/U to T200U8C/U.

5.3.3. Generalization on varying locations of UAV hubs

Since the UAV hub is deterministically located at the center in training instances, here we evaluate the generalization of SA-NNO-DRL on instances with different locations of UAV hubs. We compare SA-NNO-DRL with OR-Tools in several MURMPP instances on T200U5C with varying locations of the UAV hub. Their solutions are plotted in Fig. 7(a) and Fig. 7(b), respectively, where blue circles represent targets and the black square means the UAV hub. We draw lines with different colors to represent different UAV routes. We observe from the figures that SA-NNO-DRL still attains higher profits than OR-Tools despite the fact that the locations of UAV hubs differ from those in the training dataset. It indicates the favorable robustness of SA-NNO-DRL for different locations of UAV hubs.

5.4. Ablation study

To examine the influence of key components in the proposed NNO-DRL routing method, including the CL strategy and multi-start decoder, we conduct ablation experiments. Firstly, we remove the CL strategy from SA-NNO-DRL, resulting in a modified model referred to as SA-NNO-DRL w/o CL. Next, we replace the multi-start decoder in SA-NNO-DRL w/o CL with the original decoder from AM, creating a variant model denoted as SA-DRL. Additionally, we assess the influence of the

Table 6
Comparison results for solving individual UAV routing instances of different sizes and profit types.

Method	T20C			T50C			T20U			T50U		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Gurobi	10.58	0.00%	23.70 s	29.92	0.00%	6days	5.84	0.00%	47.04 s	16.54	0.00%	8days
Gurobi(1s)	10.52	0.65%	17.28 s	13.69	54.25%	1.03 m	5.74	1.55%	26.74 s	6.12	62.99%	1.06 m
Gurobi(10s)	10.58	0.01%	23.74 s	29.16	2.55%	5.00 m	5.83	0.01%	48.10 s	15.92	3.75%	7.07 m
Tsili	8.36	21.04%	1.67 s	21.99	26.51%	1.79 s	4.66	20.07%	1.71 s	12.12	26.75%	1.75 s
GA	9.75	7.84%	1.53 s	18.49	38.21%	5.89 m	5.51	5.54%	57.86 s	10.77	34.92%	10.36 m
Compass	10.58	0.09%	2.03 s	29.71	0.71%	18.62 s	5.82	0.23%	2.66 s	16.45	0.55%	23.09 s
AM	10.31	2.57%	0.38 s	28.44	4.96%	0.42 s	5.59	4.16%	0.37 s	15.63	5.53%	0.41 s
MADM	10.24	3.24%	0.64 s	28.14	5.95%	1.11 s	5.66	3.04%	0.64 s	15.81	4.45%	1.09 s
POMO	10.45	1.24%	0.47 s	29.10	2.75%	0.69 s	5.71	2.22%	0.51 s	15.94	3.62%	0.67 s
NNO-DRL(M=1)	10.52	0.59%	0.42 s	29.61	1.03%	0.77 s	5.75	1.42%	0.42 s	16.26	1.67%	0.75 s
NNO-DRL(M=4)	10.56	0.27%	0.60 s	29.75	0.58%	1.95 s	5.80	0.69%	0.69 s	16.39	0.94%	1.91 s

SA-based target reallocation by adjusting SA-NNO-DRL to only utilize the initial target allocation and UAV routing phase. This modified model is termed NNO-DRL. We compare the original one with SA-NNO-DRL w/o CL, SA-DRL and NNO-DRL on T40U2C/U, T60U3C/U, T80U4C/U, T100U5C/U, T150U5C/U and T200U5C/U. The hyperparameters of the DRL model in SA-DRL remain the same as the one in SA-NNO-DRL except for the learning rate on T100U5C and T200U5C, which we set to 0.00003 for better convergence in the training.

The results on instances with constant profits and uniform profits are plotted in Fig. 8 and Fig. 9, respectively. As shown, SA-NNO-DRL consistently outperforms SA-NNO-DRL w/o CL in all cases, indicating the effectiveness of the CL strategy. Additionally, SA-NNO-DRL w/o CL demonstrates superior performance compared to SA-DRL in all cases, emphasizing the contribution of the multi-start decoder. Furthermore, we observe that SA-NNO-DRL outperforms NNO-DRL in all instances, underscoring the significance of the SA-based target reallocation. In addition, we observe that the runtime of SA-NNO-DRL w/o CL and SA-DRL is almost the same as that of SA-NNO-DRL in Table 2. However, the runtime of NNO-DRL is very close to that of AM(Greedy) in Table 2 since NNO-DRL only performs the target allocation and UAV routing phases once without the iterative reallocation process.

5.5. Comparison analysis on NNO-DRL routing method

To evaluate the performance of our proposed NNO-DRL method on individual UAV routing problem (*i.e.*, orienteering problem, termed as OP), we compare our method with an exact solver (*i.e.*, Gurobi), three heuristic algorithms (*i.e.*, Tsili, GA, and Compass) and three state-of-the-art DRL-based routing methods (*i.e.*, AM, MDAM, and POMO):

1. Gurobi: We use Gurobi with different time limits including no limit, 1 s, and 10 s, denoted as Gurobi, Gurobi(1s), and Gurobi(10s) respectively.
2. Tsili [72]: A constructive heuristic for solving OP with a manually engineered function to define the node probabilities.
3. GA: A genetic algorithm for solving OP, which has also been used as a comparison algorithm in past work [18].
4. Compass [73]: The recent state-of-the-art evolutionary algorithm for solving OP incorporates the maintenance and exploration of infeasible solutions.
5. AM [18]: This method uses a Transformer-based attention model and greedy decoding strategy to generate solutions.
6. MDAM [20]: This approach extends AM by employing multiple decoders and maximizing the KL diversity between them, resulting in more diverse solutions.
7. POMO [52]: This model leverages multiple solution trajectories in parallel to achieve high-quality solutions.

We train all DRL-based solvers on problem instances with n=20 and 50 nodes (*i.e.*, targets) for all profit types, using their original settings. We train an independent model for problems of each size and profit type, and the size of the instances generated in the training process is fixed (*e.g.*, when learning the model for solving problems with n=20 and constant profits, the size and profit type of all instances used in the training process are fixed to 20 and be constant, respectively). Then, we test all DRL-based solvers on the problem with the same configuration as the training instances. Regarding the heuristic algorithms, we run them with their default parameters. For each problem configuration, we generate 1,000 problem instances. Taking problem size n=20 with constant profits and uniform profits as an example, we term them as T20C and T20U, respectively. During testing, all DRL-based solvers adopt the *Greedy* decoding strategy. We consider two versions of our NNO-DRL method, namely NNO-DRL ($\mathbb{M} = 1$) and NNO-DRL ($\mathbb{M} = 4$), where NNO-DRL ($\mathbb{M} = 1$) does not utilize instance augmentation, hence $\mathbb{M} = 1$. The performance of our NNO-DRL method and the other baselines are recorded in Table 6. Note that we report the time required to solve 1,000 instances, with DRL-based methods measured on a single GPU (GTX3090) and Gurobi/heuristic algorithms evaluated for solving 16 instances in parallel on a 16-virtual CPU system (i9-9900K).

NNO-DRL ($\mathbb{M} = 1$) demonstrates superior performance compared to Tsili, GA, and DRL-based baselines(*i.e.*, AM, MDAM, and POMO), while NNO-DRL ($\mathbb{M} = 4$) shows further improvement at the expense of slightly longer inference time. Regarding the performance against the exact solver Gurobi, our NNO-DRL in the $\mathbb{M} = 1$ and $\mathbb{M} = 4$ settings achieve gaps of less than 2% and 1%, respectively. Notably, NNO-DRL($\mathbb{M} = 4$) achieves the best objective value among all baselines except Gurobi on T50C instances. Although the Compass outperforms NNO-DRL($\mathbb{M} = 4$) in terms of objective value on T20C, T20U, and T50U, NNO-DRL($\mathbb{M} = 4$) exhibits faster inference speed, with an optimality gap between them of less than 0.5%. Overall, NNO-DRL delivers high-quality solutions within a short inference time.

6. Conclusion

This paper proposes a DRL method with SA to solve MURMPPs in an iterative two-phase framework. Specifically, it decomposes the MURMPP into the target allocation and individual UAV routing phases. In the former, we leverage SA with a well-designed shift operation to assign targets to UAVs. In the latter, we exploit an NNO-DRL model to route all UAVs in parallel. The above two processes are alternated until termination. Our method is an early attempt to handle large-scale MURMPP by solving a series of subproblems with a hybrid of DRL and meta-heuristics, drastically reducing the problem's complexity. Besides, our method provides natural scalability to the number of targets and UAVs. Results show that our method achieves good performance

in terms of solution quality and computational efficiency. Also, our method generalizes well to instances with varying numbers of targets and UAVs, and different locations of UAV hubs.

However, there are three key limitations that should be addressed in future work to further improve the performance and applicability of our SA-NNO-DRL method:

1. **Out-of-distribution performance**. In our current work, we train and test SA-NNO-DRL using instances from a uniform distribution. However, achieving good performance on out-of-distribution instances remains a challenge for DRL algorithms. As a result, our SA-NNO-DRL method may be less effective when applied to out-of-distribution instances. In the future, we will boost the robustness of SA-NNO-DRL via robust DRL technologies (*e.g.*, few-shot learning [74] and meta-learning [75]).
2. **Static instance assumption**. Our SA-NNO-DRL method currently provides target allocation schemes and UAV routes based on a defined instance without considering dynamic situations. In practice, targets may be canceled or new targets may appear, and UAVs may encounter failures or breakdowns. Therefore, it is important to adapt SA-NNO-DRL to dynamic multi-UAV mission planning scenarios. Future work will focus on incorporating dynamic elements into the planning process.
3. **Extension to additional side constraints**. Currently, our SA-NNO-DRL simplifies the MURMPP model by disregarding certain complex side constraints that are often encountered in real-life problems. For instance, there may be multiple UAV hubs, causing UAVs to start from different locations. Additionally, reconnaissance missions may have time limits, with targets assigned to specific time windows that UAVs must operate within. In our future work, we will explicitly consider these real-life constraints to enhance the applicability of SA-NNO-DRL.

In conclusion, our method is an early attempt to solve large-scale MURMPP using a hybrid of DRL and meta-heuristics, effectively addressing both computational complexity and scalability challenges. Our experimental results demonstrate that SA-NNO-DRL achieves favorable performance compared to existing methods, making it a promising solution for real-world multi-UAV mission planning scenarios.

CRediT authorship contribution statement

Mingfeng Fan: Writing – original draft, Validation, Methodology. **Huan Liu:** Writing – review & editing, Visualization. **Guohua Wu:** Writing – review & editing, Conceptualization. **Aldy Gunawan:** Writing – review & editing, Data curation. **Guillaume Sartoretti:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Sensitivity analysis

We conducted a sensitivity analysis on the SA algorithm's hyperparameters to understand their impacts on solution quality and computational efficiency, aiming to identify an optimal balance between the two. Specifically, we searched for the best hyperparameter combination across the following sets: initial temperature $T_s \in [70, 80, 90]$, the lowest temperature $T_e \in [55, 60, 65]$, Markov chain length $\mathbb{L} \in [2, 4, 6]$, and decaying rate $\sigma \in [0.85, 0.9, 0.95]$. The most representative combinations and their experimental results on the T200U5U problem are shown in Table A.7. The bolded entries indicate the parameter combination and results used in this study.

Table A.7
Results of different SA hyperparameters on T200U5U.

T_s	$\mathbb{L}$	T_e	σ	Obj.	Time
70	4	60	0.9	90.30	0.30
80	4	55	0.9	90.66	0.62
80	2	60	0.9	90.35	0.25
80	4	65	0.9	90.41	0.31
80	6	60	0.9	90.58	0.66
80	4	60	0.85	90.28	0.32
80	4	60	0.95	90.71	0.94
90	4	60	0.9	90.65	0.58
80	**4**	**60**	**0.9**	**90.62**	**0.44**

Although this particular combination does not yield the highest objective value, it provides the best trade-off by achieving high solution quality within a manageable computation time. This balance makes it well-suited for our application, effectively meeting both accuracy and efficiency requirements.

Appendix B. Statistical test

We conducted a Wilcoxon test on our method and the baselines using the experimental data on T40U2C-T200U5C and T40U2U-T200U5U to evaluate the statistical significance of performance differences. The heatmaps of p-values under the constant profit and uniform profit scenarios are shown in Figs. B.10(a) and B.10(b). This analysis highlights the robustness of our approach, showing that as problem size increases, the performance gap between our method and the baseline becomes more significant.

Appendix C. Benchmark study

We conducted a benchmark study using a publicly available TOP benchmark dataset, 'Set_33_234' (https://www.mech.kuleuven.be/en/cib/op#autotoc-item-autotoc-13), in which each instance contains 33 nodes. Notably, the TOP benchmark differs slightly from the MURMPP defined in this paper: in the former, agents start from a starting node and end at a different ending node, whereas, in our problem MURMPP, the UAVs both start and return to the same node. To align with our problem setup, we removed the final node (ending node) in this dataset, resulting in modified instances with 32 nodes, with the first node designated as the UAV hub. We select 10 instances from 'Set_33_234', each involving two UAVs, to evaluate our method and the baselines. The experimental results, presented in Table C.8, show that Gurobi performs best, with our method performing comparably to OR-Tools and outperforming other learning-based methods (AM and GAMMA). It is worth noting that the distribution of the benchmark instances differs significantly from that of our training data, which can affect our method's performance. The degradation in performance on out-of-distribution data is a common limitation of deep learning methods.

Data availability

Data will be made available on request.

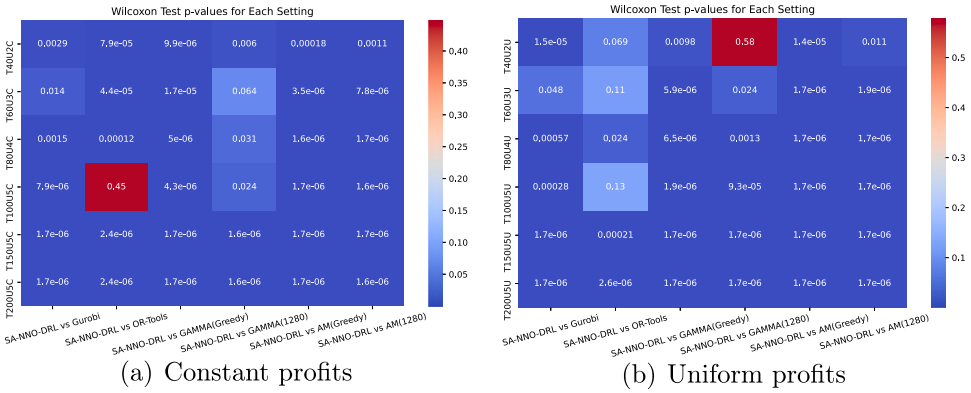


Fig. B.10. Wilcoxon test on MURMPP instances with different problem sizes and profit types.

Table C.8
Results on TOP benchmark instances.

Instance	Gurobi	OR-Tools	GAMMA (Greedy)	GAMMA (1280)	AM (Greedy)	AM(1280)	SA-NNO-DRL
p3.2.j	**500**	**500**	410	470	420	490	**500**
p3.2.l	**590**	570	500	580	530	580	570
p3.2.m	**610**	590	500	**610**	560	580	600
p3.2.n	**650**	610	510	630	510	610	630
p3.2.o	**680**	630	580	640	530	670	660
p3.2.p	**710**	700	610	690	680	680	690
p3.2.q	**750**	**750**	610	710	710	710	720
p3.2.r	**780**	770	640	750	680	740	770
p3.2.s	**800**	780	720	760	750	750	770
p3.2.t	**800**	**800**	730	780	750	750	**800**

References

[1] D. Chen, Research on traffic flow prediction in the big data environment based on the improved RBF neural network, IEEE Trans. Ind. Inform. 13 (4) (2017) 2000–2008.

[2] M. Elloumi, R. Dhaou, B. Escrig, H. Idoudi, L.A. Saidane, Monitoring road traffic with a UAV-based system, in: 2018 IEEE Wireless Communications and Networking Conference, WCNC, IEEE, 2018, pp. 1–6.

[3] W. Zhang, K. Song, X. Rong, Y. Li, Coarse-to-fine UAV target tracking with deep reinforcement learning, IEEE Trans. Autom. Sci. Eng. 16 (4) (2019) 1522–1530, http://dx.doi.org/10.1109/TASE.2018.2877499.

[4] B.D. Song, K. Park, J. Kim, Persistent UAV delivery logistics: MILP formulation and efficient heuristic, Comput. Ind. Eng. 120 (2018) 418–428.

[5] Z. Junwei, Z. Jianjun, Study on multi-UAV task clustering and task planning in cooperative reconnaissance, in: 2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics, Vol. 2, IEEE, 2014, pp. 392–395.

[6] V.S. Nagmode, S. Rajbhoj, An IoT platform for vehicle traffic monitoring system and controlling system based on priority, in: 2017 International Conference on Computing, Communication, Control and Automation, ICCUBEA, IEEE, 2017, pp. 1–5.

[7] R. Gedik, E. Kirac, A.B. Milburn, C. Rainwater, A constraint programming approach for the team orienteering problem with time windows, Comput. Ind. Eng. 107 (2017) 178–195.

[8] C. Archetti, D. Feillet, A. Hertz, M.G. Speranza, The capacitated team orienteering and profitable tour problems, J. Oper. Res. Soc. 60 (2009) 831–842.

[9] S.E. Butt, D.M. Ryan, An optimal solution procedure for the multiple tour maximum collection problem using column generation, Comput. Oper. Res. 26 (4) (1999) 427–441.

[10] M. Keshtkaran, K. Ziarati, A. Bettinelli, D. Vigo, Enhanced exact solution methods for the team orienteering problem, Int. J. Prod. Res. 54 (2) (2016) 591–601.

[11] R. El-Hajj, D.-C. Dang, A. Moukrim, Solving the team orienteering problem with cutting planes, Comput. Oper. Res. 74 (2016) 21–30.

[12] L. Ke, C. Archetti, Z. Feng, Ants can solve the team orienteering problem, Comput. Ind. Eng. 54 (3) (2008) 648–665.

[13] H. Tang, E. Miller-Hooks, A tabu search heuristic for the team orienteering problem, Comput. Oper. Res. 32 (6) (2005) 1379–1407.

[14] C. Archetti, A. Hertz, M.G. Speranza, Metaheuristics for the team orienteering problem, J. Heuristics 13 (1) (2007) 49–76.

[15] D.-C. Dang, R.N. Guibadj, A. Moukrim, An effective PSO-inspired algorithm for the team orienteering problem, European J. Oper. Res. 229 (2) (2013) 332–344.

[16] S.-W. Lin, Solving the team orienteering problem using effective multi-start simulated annealing, Appl. Soft Comput. 13 (2) (2013) 1064–1073.

[17] Y. Hu, Y. Yao, W.S. Lee, A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs, Knowl.-Based Syst. 204 (2020) 106244.

[18] W. Kool, H. Van Hoof, M. Welling, Attention, learn to solve routing problems, in: The 6th International Conference on Learning Representations, ICLR, 2018.

[19] X. Chen, Y. Tian, Learning to perform local rewriting for combinatorial optimization, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 6281–6292.

[20] L. Xin, W. Song, Z. Cao, J. Zhang, Multi-decoder attention model with embedding glimpse for solving vehicle routing problems, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 13, 2021, pp. 12042–12049.

[21] L. Xin, W. Song, Z. Cao, J. Zhang, Step-wise deep learning models for solving routing problems, IEEE Trans. Ind. Inform. 17 (7) (2020) 4861–4871.

[22] Y. Wu, W. Song, Z. Cao, J. Zhang, A. Lim, Learning improvement heuristics for solving routing problems, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[23] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, L. Song, Learning combinatorial optimization algorithms over graphs, in: Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 6351–6361.

[24] C. Zhang, W. Song, Z. Cao, J. Zhang, P.S. Tan, X. Chi, Learning to dispatch for job shop scheduling via deep reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1621–1632.

[25] S. Luo, L. Zhang, Y. Fan, Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning, IEEE Trans. Autom. Sci. Eng. (2021) 1–19, http://dx.doi.org/10.1109/TASE.2021.3104716.

[26] Y. He, L. Xing, Y. Chen, W. Pedrycz, L. Wang, G. Wu, A generic Markov decision process model and reinforcement learning method for scheduling agile earth observation satellites, IEEE Trans. Syst. Man, Cybern. Syst. (2020).

[27] F. Wang, X. Wang, S. Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, Inform. Sci. 602 (2022) 298–312.

[28] E.J. Forsmo, E.I. Grøtli, T.I. Fossen, T.A. Johansen, Optimal search mission with unmanned aerial vehicles using mixed integer linear programming, in: 2013 International Conference on Unmanned Aircraft Systems, ICUAS, IEEE, 2013, pp. 253–259.

[29] J. Cui, Y. Liu, A. Nallanathan, The application of multi-agent reinforcement learning in UAV networks, in: 2019 IEEE International Conference on Communications Workshops, ICC Workshops, IEEE, 2019, pp. 1–6.

[30] P. Sankaran, K. McConky, M. Sudit, H. Ortiz-Peña, GAMMA: Graph attention model for multiple agents to solve team orienteering problem with multiple depots, IEEE Trans. Neural Netw. Learn. Syst. (2022).

[31] H. Guo, Z. Liu, R. Shi, W.-Y. Yau, D. Rus, Cross-entropy regularized policy gradient for multirobot nonadversarial moving target search, IEEE Trans. Robot. 39 (4) (2023) 2569–2584, http://dx.doi.org/10.1109/TRO.2023.3263459.

[32] W. Yao, N. Qi, N. Wan, Y. Liu, An iterative strategy for task assignment and path planning of distributed multiple unmanned aerial vehicles, Aerosp. Sci. Technol. 86 (2019) 455–464.
[33] H. Liu, X. Li, G. Wu, M. Fan, R. Wang, L. Gao, W. Pedrycz, An iterative two-phase optimization method based on divide and conquer framework for integrated scheduling of multiple UAVs, IEEE Trans. Intell. Transp. Syst. 22 (9) (2020) 5926–5938.
[34] X. Mao, G. Wu, M. Fan, Z. Cao, W. Pedrycz, DL-DRL: A double-level deep reinforcement learning approach for large-scale task scheduling of multi-UAV, IEEE Trans. Autom. Sci. Eng. (2024).
[35] Q. Deng, B.F. Santos, R. Curran, A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization, European J. Oper. Res. 281 (2) (2020) 256–273.
[36] B. Alidaee, H. Wang, F. Landram, A note on integer programming formulations of the real-time optimal scheduling and flight path selection of UAVs, IEEE Trans. Control Syst. Technol. 17 (4) (2009) 839–843.
[37] T. Li, J. Jiang, Z. Zhen, C. Gao, Mission planning for multiple UAVs based on ant colony optimization and improved dubins path, in: 2016 IEEE Chinese Guidance, Navigation and Control Conference, CGNCC, IEEE, 2016, pp. 954–959.
[38] J. Tian, L. Shen, Y. Zheng, Genetic algorithm based approach for multi-UAV cooperative reconnaissance mission planning problem, in: International Symposium on Methodologies for Intelligent Systems, Springer, 2006, pp. 101–110.
[39] X. Bai, W. Yan, S.S. Ge, M. Cao, An integrated multi-population genetic algorithm for multi-vehicle task assignment in a drift field, Inform. Sci. 453 (2018) 227–238.
[40] Y. Liu, R. Song, R. Bucknall, X. Zhang, Intelligent multi-task allocation and planning for multiple unmanned surface vehicles (USVs) using self-organising maps and fast marching method, Inform. Sci. 496 (2019) 180–197.
[41] Y. Wu, T. Liang, J. Gou, C. Tao, H. Wang, Heterogeneous mission planning for multiple uav formations via metaheuristic algorithms, IEEE Trans. Aerosp. Electron. Syst. 59 (4) (2023) 3924–3940.
[42] G. Xu, X. Kang, H. Yang, Y. Wu, W. Liu, J. Cao, Y. Liu, Distributed multi-vehicle task assignment and motion planning in dense environments, IEEE Trans. Autom. Sci. Eng. (2023).
[43] G. Xu, Y. Wu, S. Tao, Y. Yang, T. Liu, T. Huang, H. Wu, Y. Liu, Multi-robot task allocation and path planning with maximum range constraints, 2024, arXiv preprint arXiv:2409.06531.
[44] H. Guo, Q. Peng, Z. Cao, Y. Jin, DRL-searcher: A unified approach to multirobot efficient search for a moving target, IEEE Trans. Neural Netw. Learn. Syst. 35 (3) (2024) 3215–3228, http://dx.doi.org/10.1109/TNNLS.2023.3274667.
[45] A. Hottung, K. Tierney, Neural large neighborhood search for the capacitated vehicle routing problem, 2019, arXiv preprint arXiv:1911.09539.
[46] P.R. d O Costa, J. Rhuggenaath, Y. Zhang, A. Akcay, Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning, in: Asian Conference on Machine Learning, PMLR, 2020, pp. 465–480.
[47] M. Kim, J. Park, et al., Learning collaborative policies to solve NP-hard routing problems, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 10418–10430.
[48] Y. Ma, J. Li, Z. Cao, W. Song, L. Zhang, Z. Chen, J. Tang, Learning to iteratively solve routing problems with dual-aspect collaborative transformer, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 11096–11107.
[49] C. Fang, Z. Han, W. Wang, E. Zio, Routing UAVs in landslides monitoring: A neural network heuristic for team orienteering with mandatory visits, Transp. Res. Part E: Logist. Transp. Rev. 175 (2023) 103172.
[50] I. Bello, H. Pham, Q.V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, in: International Conference on Learning Representations(Workshop), 2017.
[51] M. Nazari, A. Oroojlooy, L. Snyder, M. Takác, Reinforcement learning for solving the vehicle routing problem, in: Advances in Neural Information Processing Systems, vol. 31, 2018, pp. 9839–9849.
[52] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, S. Min, Pomo: Policy optimization with multiple optima for reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 21188–21198.
[53] M. Kim, J. Park, J. Park, Sym-nco: Leveraging symmetricity for neural combinatorial optimization, in: Advances in Neural Information Processing Systems, 2022.
[54] F. Guo, Q. Wei, M. Wang, Z. Guo, S.W. Wallace, Deep attention models with dimension-reduction and gate mechanisms for solving practical time-dependent vehicle routing problems, Transp. Res. Part E: Logist. Transp. Rev. 173 (2023) 103095.
[55] B. Li, G. Wu, Y. He, M. Fan, W. Pedrycz, An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem, IEEE/CAA J. Autom. Sin. 9 (7) (2022) 1115–1138.
[56] O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, in: Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 2692–2700.
[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, vol. 30, 2017.
[58] H. Ye, J. Wang, Z. Cao, H. Liang, Y. Li, Deepaco: neural-enhanced ant systems for combinatorial optimization, Adv. Neural Inf. Process. Syst. 36 (2024).
[59] M. Kim, S. Choi, J. Son, H. Kim, J. Park, Y. Bengio, Ant colony sampling with gflownets for combinatorial optimization, 2024, arXiv preprint arXiv:2403.07041.
[60] H. Ye, J. Wang, H. Liang, Z. Cao, Y. Li, F. Li, Glop: Learning global partition and local construction for solving large-scale routing problems in real-time, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, No. 18, 2024, pp. 20284–20292.
[61] X. Chen, Y. Li, Y. Yang, L. Zhang, S. Li, G. Pan, Extnco: A fine-grained divide-and-conquer approach for extending nco to solve large-scale traveling salesman problem, 2023, Available At SSRN 4679437.
[62] Z. Zheng, C. Zhou, T. Xialiang, M. Yuan, Z. Wang, Udc: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems, 2024, arXiv preprint arXiv:2407.00312.
[63] X. Chen, K. He, Exploring simple siamese representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15750–15758.
[64] Y. Ma, J. Li, Z. Cao, W. Song, H. Guo, Y. Gong, Y.M. Chee, Efficient neural neighborhood search for pickup and delivery problems, in: International Joint Conference on Artificial Intelligence, IJCAI, 2022.
[65] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), 2016, arXiv preprint arXiv:1606.08415.
[66] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, D. Krishnan, Supervised contrastive learning, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 18661–18673.
[67] T. Teshima, I. Sato, M. Sugiyama, Few-shot domain adaptation by causal mechanism transfer, in: International Conference on Machine Learning, PMLR, 2020, pp. 9458–9469.
[68] D. Zhang, F. Nan, X. Wei, S. Li, H. Zhu, K. McKeown, R. Nallapati, A. Arnold, B. Xiang, Supporting clustering with contrastive learning, in: Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL, 2021.
[69] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, 2014.
[70] L. Gurobi Optimization, Gurobi Optimizer Reference Manual, URL https://www.Gurobi.Com/Products/Gurobi-Optimizer/.
[71] P. L., F. V., OR-Tools, URL https://developers.google.com/optimization/.
[72] T. Tsiligirides, Heuristic methods applied to orienteering, J. Oper. Res. Soc. 35 (1984) 797–809.
[73] G. Kobeaga, M. Merino, J.A. Lozano, An efficient evolutionary algorithm for the orienteering problem, Comput. Oper. Res. 90 (2018) 42–59.
[74] Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: A survey on few-shot learning, ACM Comput. Surv. (Csur) 53 (3) (2020) 1–34.
[75] T. Hospedales, A. Antoniou, P. Micaelli, A. Storkey, Meta-learning in neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (9) (2021) 5149–5169.