A Three-Layer Reinforcement Learning-based Approach for Dynamic Task Allocation Under Multiple Task Resource Constraints

Shuo Wang
Zhengzhou University
Zhengzhou, China
wangyz2000@gs.zzu.edu.cn

Yuzhen Zhang*
Zhengzhou University
Zhengzhou, China
yzzhang@zzu.edu.cn

Hao Su
Zhengzhou University
Zhengzhou, China
iesuhao@zzu.edu.cn

Yazhou Hu
Zhengzhou University
Zhengzhou, China
hooasia1990@outlook.com

Pei Lv
Zhengzhou University
Zhengzhou, China
ielvpei@zzu.edu.cn

ABSTRACT

This paper addresses the **S**ingle-**T**ask robot, **S**ingle-**R**obot task, **T**ime-extended **A**ssignment (ST-SR-TA) problem in Multi-Robot Task Allocation (MRTA), focusing on dynamic robot arrivals, diverse task requirements, and heterogeneous robot resources. However, achieving effective allocation in such dynamic environments to optimize cumulative rewards remains a challenge. Classical exact algorithms like dynamic programming are limited to small-scale tasks due to exponential complexity, while heuristic methods typically operate iteratively, requiring re-execution when the environment changes and thus struggling to adapt to dynamic scenarios. To overcome these limitations, we propose a three-layer reinforcement learning method based on attention mechanisms. Specifically, we decompose the task allocation problem with multiple task resource constraints into two sub-problems: capability matching and sequence optimization. We then exploit the attribute and spatial relationships between robots and tasks to design a hierarchical RL strategy by introducing two attention mechanisms. This strategy addresses the whole problem progressively across three layers: *Global Allocation* performs task–robot matching from a global perspective to generate a candidate task pool, *Individual Selection* determines executable tasks for each robot based on its real-time state, and *Sequence Optimization* refines the task execution order according to spatial relationships to minimize path cost. Experimental results show that the proposed method significantly improves the task reward compared to other approaches. Moreover, the method demonstrates few-shot and zero-shot generalization to new task allocation scenarios, providing an efficient and practical solution.

KEYWORDS

Multi-Robot Task Allocation; Reinforcement Learning; Dynamic Environments; Multiple Task-Resource Constraints; Attention Mechanism

*Corresponding author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/CPZL2237

ACM Reference Format:
Shuo Wang, Yuzhen Zhang, Hao Su, Yazhou Hu, and Pei Lv. 2026. A Three-Layer Reinforcement Learning-based Approach for Dynamic Task Allocation Under Multiple Task Resource Constraints. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/CPZL2237

1 INTRODUCTION

Multi-Robot Task Allocation (MRTA) is a fundamental combinatorial optimization problem [8, 14], aiming to efficiently assign heterogeneous tasks to robots with diverse capabilities under complex resource constraints, to maximize global objectives such as resource utilization and overall task reward. It serves as a cornerstone in applications including intelligent traffic dispatching, collaborative manufacturing, and emergency response [15, 24, 33]. Within this domain, we focus on the ST-SR-TA problem [18], which is commonly formulated as an extension of the Vehicle Routing Problem (VRP) [27] or the Multiple Traveling Salesman Problem (mTSP) [2, 13]. While exact algorithms (e.g., dynamic programming) guarantee optimality, their computational cost grows exponentially with problem scale. Meta-heuristic approaches, on the other hand, rely heavily on handcrafted rules and struggle to adapt to dynamic environments where robots continuously enter or leave the system. Furthermore, both mTSP and VRP-based formulations primarily minimize path cost, and even their extensions such as the Capacitated VRP (CVRP) [10] consider only single-resource constraints, failing to capture multi-resource supply–demand matching that frequently arises in real-world MRTA scenarios.

Reinforcement learning (RL) has gained growing interest in MRTA due to its ability to autonomously learn decision-making strategies in dynamic and uncertain environments. For example, Yuan et al. [29] developed an improved deep Q-network for MRTA in robotic mobile fulfillment systems, while Elfakharany et al. [9] introduced a decentralized deep RL framework that jointly optimizes task allocation and navigation in homogeneous robot teams. In addition, attention mechanisms have been increasingly integrated into RL architectures to better capture task–robot correlations and improve allocation performance [1, 11, 19]. However, these methods remain constrained by fixed-scale setups and simplistic allocation criteria. Most assume a predefined team size and static network,

with strategies for dynamic teams often limited to basic activation/deactivation [11]. Furthermore, allocation decisions are often oversimplified, relying on single factors such as spatial location or compatibility [11, 17] while ignoring multi-resource requirements.

To address these limitations, we propose a novel three-layer reinforcement learning framework that explicitly decouples the ST-SR-TA problem into two complementary sub-problems: "Who does What" (task-robot capability matching) and "In Which Order" (task execution sequencing). This decomposition allows us to jointly optimize for maximum task reward and minimal path cost. The framework operates hierarchically: the top layer performs global task-robot matching, the middle layer selects tasks based on individual robot states, and the bottom layer optimizes the final execution sequence. An integrated attention mechanism enables dynamic perception of heterogeneous features, endowing the model with superior generalization to unseen environments. Our main contributions are fourfold.

- We decompose the allocation problem under multiple task resource constraints into two sub-problems and design a three-layer reinforcement learning approach. This approach further incorporates an attention mechanism to effectively accommodate dynamic scenarios with variable inputs.
- We introduce a "Global Allocation – Individual Selection" strategy consisting of two modules: global matching generates a task candidate pool to avoid local optimality, and individual selection assigns each robot an appropriate number of tasks from the pool according to its state, thus striving for the best possible reward.
- We propose a task sequence optimization module that refines the execution order of tasks selected by the upper layer. The module learns from the spatial distribution of dynamically output, variable-sized task subsets and reduces the robots' travel costs through sequence optimization.
- Our method demonstrates robust zero- and few-shot generalization: in zero-shot settings, it achieves approximately 80% of the performance of 1000-step from-scratch training; in few-shot adaptation, it reaches the same 1000-step performance within only 200–300 training steps.

2 RELATED WORK

Multi-robot task allocation (MRTA) is commonly categorized using the three-dimensional taxonomy proposed by Korsah et al. [18], which classifies MRTA problems along the axes of task load (ST/MT), robot–task mapping (SR/MR), and allocation horizon (IA/TA). Among the resulting eight subtypes, the ST-SR-TA setting—where each robot handles one task at a time, each task is performed by one robot, and allocation is planned over a time horizon—has been extensively studied due to its relevance to real-world robotic systems.

Conventional approaches to MRTA are broadly classified into market-auction-based and optimization-based methods. Market-auction mechanisms, inspired by economic transactions, enable robots to bid for tasks under predefined rules to maximize collective utility. A prominent example is the Consensus-Based Bundle Algorithm (CBBA) [5], which employs iterative local communication to reach conflict-free allocations. However, due to its greedy decision-making mechanism, CBBA often converges to locally optimal solutions, particularly when the number of tasks significantly exceeds that of robots [4, 21]. Furthermore, the iterative consensus process requires multiple communication rounds, leading to delayed convergence and limiting applicability in highly dynamic or real-time scenarios [7].

In contrast, optimization-based methods predominantly rely on metaheuristic algorithms—such as Genetic Algorithms (GA) [20], Particle Swarm Optimization (PSO) [22], Simulated Annealing (SA) [3], and Symbiotic Organisms Search (SOS) [25]—to search for near-optimal allocations by emulating natural or biological processes. While these methods perform effectively in static settings, they struggle to adapt when task or robot attributes change dynamically. Some studies attempt to incrementally refine solutions using mechanisms like partial crossover or mutation in GA [20, 32], or by restarting the optimization process entirely [30]. Nevertheless, such strategies remain computationally expensive, as each environmental change necessitates repeated iterations, rendering these methods unsuitable for rapidly evolving operational contexts.

Deep Reinforcement Learning (DRL) has emerged as a promising alternative for combinatorial optimization, owing to its capacity for learning adaptive decision policies through environmental interaction. Recent studies have extended DRL to MRTA with increasingly sophisticated designs. Zhou et al. [31] developed a hierarchical reinforcement learning framework to address scalability and dynamism in large-scale robotic mobile fulfillment systems, while Xu et al. [28] proposed MATD3-TORA to optimize energy-aware task offloading in UAV swarms. Attention mechanisms have further enhanced relational reasoning in these settings: Park et al. [19] employed cross-attention to model robot-robot coordination and task relevance, whereas Agrawal et al. [1] and Gong et al. [11] utilized attention to encode heterogeneous robot and task representations, improving generalization in warehouse logistics and dynamic team sizing.

Despite these advances, existing methods remain limited in several key aspects. They typically assume a fixed robot team, failing to accommodate newly joining robots during operation. Moreover, their allocation criteria often rely narrowly on spatial relationships [16, 17] or intrinsic attribute matching [11], treating other critical factors—such as multi-resource requirements—as constraints rather than integral components of the optimization objective. In contrast, our approach explicitly addresses these shortcomings through a structured decomposition of the allocation process and a flexible attention-based architecture capable of handling dynamic team changes and multi-faceted task characteristics.

3 PROBLEM SETUP

We investigate a multi-task resource allocation and transportation problem, where numerous tasks are distributed across different locations and heterogeneous robots continuously arrive over time. The objective is to rationally allocate tasks to maximize overall reward. An episode terminates when robot resources are exhausted, all tasks are completed, or a predefined time limit is reached, during which multiple allocations may occur. We formulate the problem as a Markov Decision Process (MDP), where the system state encompasses both robot and task information. Our method outputs actions

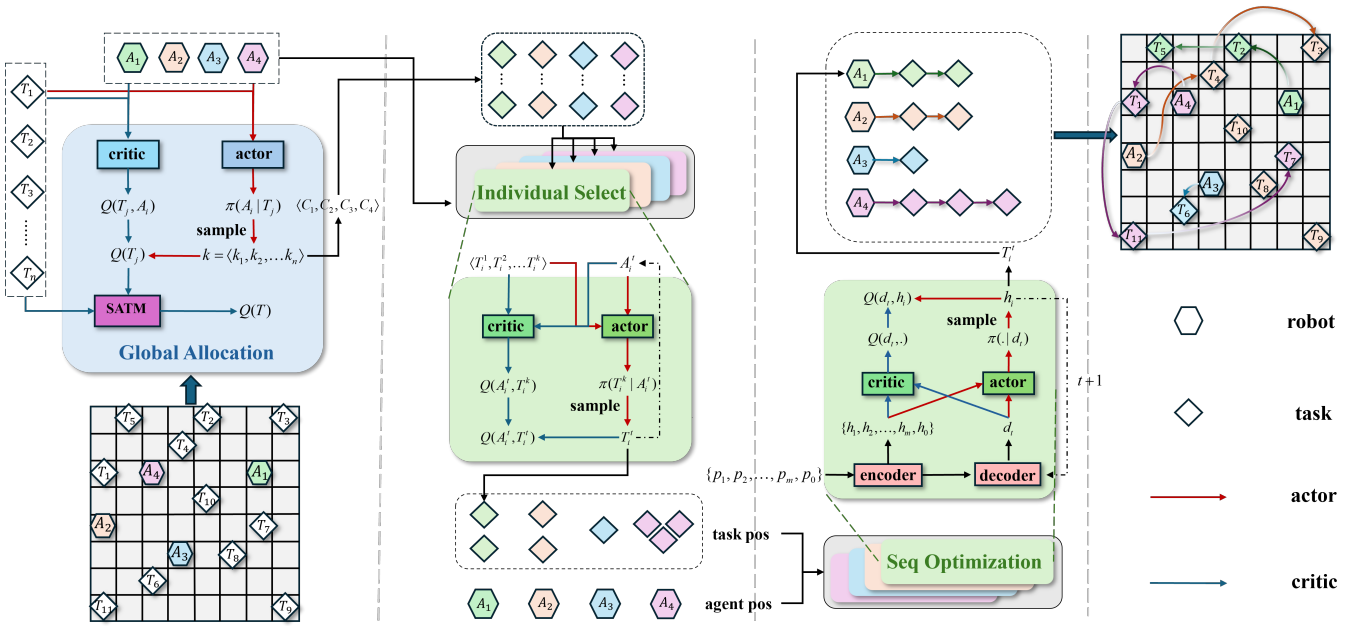


Figure 1: Overview of the proposed three-layer allocation approach. The Global Allocation layer generates candidate task pools for each robot based on task and robot attributes. The Individual Selection layer then selects pending tasks, and the Sequence Optimization layer refines their execution order, producing the final task execution sequence.

in the form of sequential task queues, which are then executed by the robots.

Robot: Let $A = \{A_1, A_2, \ldots, A_m\}$ denote the set of robots in the environment. Each robot A_i is defined by a tuple $\langle a_i, c_i, p_i, K_i \rangle$, where $\langle a_1^i, a_2^i, \ldots, a_k^i \rangle$ denotes its attributes (e.g., carried resources); c_i is the activation cost; $p_i = (x_i, y_i)$ represents its coordinates; $K_i \in \{0, 1\}$ indicates the robot's availability (1: allocated).

Task: Let $T = \{T_1, T_2, \ldots, T_n\}$ denote the set of tasks in the environment. Each task T_j is defined by a tuple $\langle t_j, r_j, q_j, Q_j, S_j \rangle$, where $\langle t_j^1, t_j^2, \ldots, t_j^k \rangle$ represents its attributes (e.g., resource requirements); r_j denotes the completion reward; $q_j = (x_j, y_j)$ denotes the task location; $Q_j \in \{0, 1\}$ indicates completion status (1: completed); $S_j \in \{0, 1\}$ indicates allocation status (1: allocated).

We define a binary assignment variable $x_{ij} \in \{0, 1\}$ indicating whether task T_j is assigned to robot A_i (1 if assigned, 0 otherwise). Our objective is to output an ordered task allocation sequence $L = \langle L_1, \ldots, L_m \rangle$, where L_i denotes the task execution sequence of robot A_i. The allocation is subject to the following constraints:

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall j = 1, \ldots, n. \tag{1}$$

Equation (1) ensures that each task is assigned to at most one robot.

$$x_{ij} T_{j,k} \leq A_{i,k}, \quad \forall i = 1, \ldots, m, \ \forall j = 1, \ldots, n, \ \forall k = 1, \ldots, K. \tag{2}$$

Equation (2) enforces the feasibility/resource constraint: if task T_j is assigned to robot A_i, then the robot must satisfy the task requirement on every resource dimension k.

$$A_{i,k} \leftarrow A_{i,k} - x_{ij} T_{j,k}, \quad \forall k = 1, \ldots, K. \tag{3}$$

Equation (3) specifies the resource update rule: when a task is assigned and satisfies (2), the corresponding resources are consumed accordingly; if a robot cannot satisfy all requirements of its assigned task during execution, it is regarded as resource-exhausted (i.e., the remaining resources are set to zero as an exhaustion/exit rule in the environment). We aim to maximize the total reward, as formulated in Equation (4):

$$\text{Return} = \sum_{i=1}^{m} R_i, \quad R_i = \sum_{j \in L_i} r_j - c_i - d_i \tag{4}$$

The goal is to ensure that robots have sufficient and fully utilizable resources for their assigned tasks, thereby reducing the total utilization costs c_i and the movement costs d_i incurred in task completion, while increasing the obtained task rewards r_j.

4 METHODOLOGY

As shown in Fig. 1, our method takes the scenario's state information as input and outputs an execution strategy for each robot, specifying the order in which each robot executes its assigned tasks. We set the maximum number of robots allowed in the environment to M, resulting in an action space of dimension $n \times M$, where n denotes the number of tasks. As illustrated in Fig. 2, when the resources of active robots are insufficient, robots leave the scenario. If robots are available in the backup queue, they are deployed to replenish the task environment.

Once a robot starts executing a task, its status is marked as unallocatable ($K_i = 0$). We adopt the Soft Actor-Critic (SAC) algorithm [12] as the base reinforcement learning framework, and employ its offline variant [6] to address the discrete action space

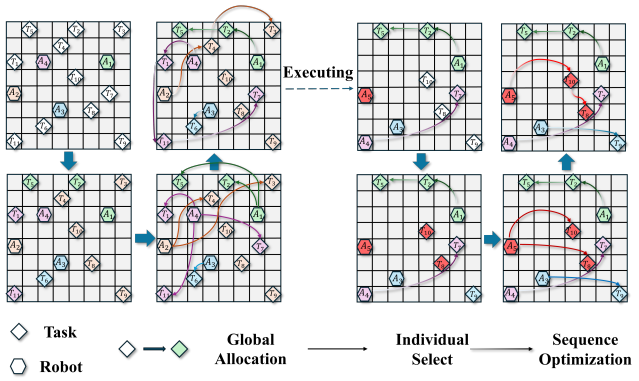


Figure 2: After the initial allocation enters the execution stage, Robot 2 leaves due to resource depletion, while Robot 5 joins the environment. Meanwhile, Robot 3 finishes all assigned tasks and remains available. At this point, Robots 3 and 5 participate as new robots in the next round of allocation.

inherent in the task allocation problem. During an episode, the allocation process will be reinitiated when any of the following conditions is satisfied:

(i) **Insufficient resources.** When a robot lacks sufficient resources to complete its remaining tasks and there exists a backup robot, the backup robot is dispatched as a replacement. The unexecuted tasks of the replaced robot are reset to the "to be allocated" state, expressed as $Q_j = 0 \implies S_j = 0$, where $Q_j = 0$ indicates that the task is uncompleted and $S_j = 0$ means it remains unallocated.

(ii) **Task completion.** When a robot finishes all its assigned tasks but still has remaining resources, its status is updated from unallocatable ($K_i = 0$) to allocable ($K_i = 1$), i.e., $K_i = 0 \implies K_i = 1$. A new allocation round is triggered when there exist unallocated tasks ($S_j = 0$) and allocatable robots ($K_i = 1$).

All allocation processes are divided into three stages: **Global Allocation**, **Individual Selection**, and **Sequence Optimization**. In each stage, an actor network generates the allocation strategy, while a critic network evaluates the corresponding allocation value.

4.1 Global Allocation

The Global Allocation stage assigns all unallocated tasks to suitable robots based on task-robot compatibility. As shown in Fig. 3, the actor network generates strategies by first employing two embedding functions: a robot embedding function $f^a : \mathbb{R}^{m\times k} \to \mathbb{R}^{m\times d}$, and a task embedding function $f^t : \mathbb{R}^{n\times k} \to \mathbb{R}^{n\times d}$. The embedding function structure consists of a double-layer linear transformation, GELU activation, and LayerNorm normalization, and is defined as:

$$f(x) = W_2 \cdot \mathrm{LN}\left(\mathrm{GELU}(W_1 x + b_1)\right) + b_2 \tag{5}$$

Subsequently, the attributes of robots and tasks are embedded into high-dimensional vectors, $E_i^a = f^a(A_i)$ and $E_j^t = f^t(T_j)$, respectively. The matching score is then obtained via a scaled dot-product attention module:

$$\text{score} = \frac{(E_i^a)^\top E_j^t}{\sqrt{d}} \tag{6}$$

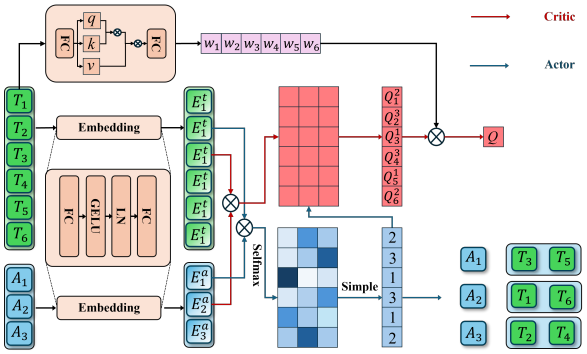


Figure 3: Global Allocation.

By combining this with the allocable flags $S = \langle S_1, S_2, \ldots, S_n \rangle$, the probability of task T_j being allocated to robot A_i is calculated using Softmax:

$$\pi(A_i \mid T_j) = \mathrm{Softmax}(\text{score}, S) \tag{7}$$

The allocation result k_i for each allocable task is sampled from $\pi(A_i \mid T_j)$, and the total allocation result is $k = \langle k_1, k_2, \ldots, k_n \rangle$ (if $k_j = i$, task T_j is allocated to robot A_i). From the assignment vector k, we form per-robot candidate pools $C = \langle C_1, \ldots, C_m \rangle$, where $C_i = \{\, j \mid k_j = i \,\}$.

Similar to the actor network, we use embedding functions f^o and f^p, analogous to those in the policy network, to encode the attributes of robots and tasks into high-dimensional vectors $E_i^o = f^o(A_i)$ and $E_j^p = f^p(T_j)$. The matching score is then computed via dot product, serving as the value weight for assigning T_j to A_i:

$$Q(T_j, \cdot) = (E_i^o)^\top E_j^p \tag{8}$$

At this stage, the matrix has dimensions $N \times M$. The allocation value $Q(k)$ for each task is obtained by combining this matrix with the sampling result k from the actor network. Since global allocation requires interaction with the entire environment, a single aggregated Q-value is needed to represent the global allocation reward. Inspired by the QMIX approach for aggregating multi-agent Q-values [23], we introduce a novel task-level Q-value aggregation method, termed Self-Attention Task MIX (SATM). This method leverages a self-attention module to model inter-task dependencies, thereby replacing traditional weighted summation. SATM not only improves aggregation accuracy by capturing correlations among tasks, but also dynamically generates parameter weights based on the task set size, allowing Q-value integration for task subsets of arbitrary cardinality. The actor network loss in the global allocation module is defined as:

$$\mathcal{L}_{\text{actor}} = -\alpha \cdot \sum \pi \cdot \log \pi - \mathrm{SATM}\left(\sum \pi \cdot Q(T, k)\right) \tag{9}$$

The loss of the critic network is:

$$\mathcal{L}_{\text{critic}} = \frac{1}{2} \left\| r - \mathrm{SATM}\left(Q(T, k)\right) \right\|^2 \tag{10}$$

4.2 Individual Selection

After the global allocation stage, each robot is assigned a candidate task pool, and we need to select a subset of tasks with an appropriate

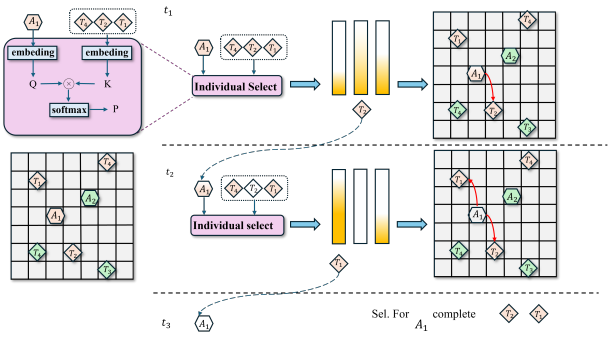


Figure 4: Individual Selection.

quantity and diverse requirements from this pool for execution. Since a robot's resources are updated after each assignment, the network module treats it as a newly defined instance, and the selection module should likewise possess the capability to allocate tasks to these "new" robots. As shown in Fig. 4, The state of robot A_i at the t-th selection is denoted by A_i^t. An embedding function maps this state into a vector e_i^t, while each task T_j in its candidate pool is mapped into a vector g_j. Unlike the global allocation stage, which assigns available tasks to robots, this stage focuses on the sequential selection of tasks by robots from their candidate pools. The selection strategy is then defined as:

$$\pi(T^t \mid e_i^t) = \text{Softmax}\left(\frac{g_j^\top e_i^t}{\sqrt{d}}\right) \tag{11}$$

Through sampling, the most suitable task T^t at time t is selected, and the robot's state is updated as $A_i^{t+1} = A_i^t - f(T^t)$. This iterative process continues until the robot either exhausts its resources or all tasks in its pre-selection pool have been chosen. Task selection then shifts to the next robot. After all robots complete their selections, an execution sequence $l = \langle l_1, l_2, \ldots, l_m \rangle$ is formed, which specifies task assignments across robots but does not define their execution order. The final task execution order is determined in the subsequent path optimization stage. The reward obtained by each individual during task selection is:

$$R_i = \sum_{j \in a_i} r_j - c_i \tag{12}$$

The rewards obtained by all allocated robots are aggregated to form the reward for the upper-layer global allocation.

$$R = \sum_{i=1}^{m} R_i \tag{13}$$

4.3 Sequence Optimization

After obtaining the set of tasks to be executed from the individual selection stage, we further optimize the execution order to significantly reduce travel distance costs. For the dynamically changing task sets generated by the upper-level allocation, we design a path optimization module inspired by the Pointer Network (Pointer-Net) method [26]. This module models the sequential decision-making process using a pointer network and incorporates an attention

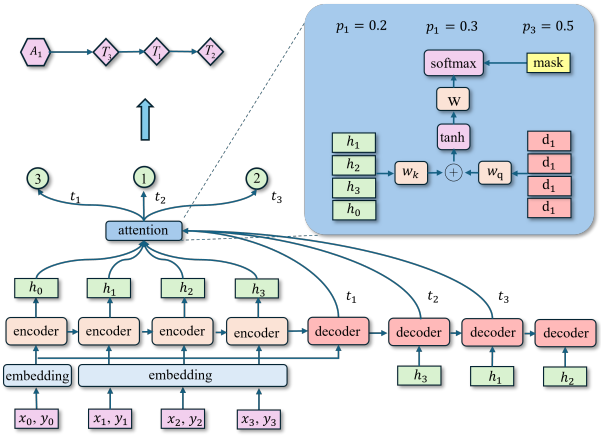


Figure 5: Sequence Optimization.

mechanism based on reinforcement learning, thereby enabling efficient planning of task execution sequences, as illustrated in Fig. 5.

First, the module encodes features of the robot's current position and the coordinates of allocated tasks. Let the robot's current position be $p_0 \in \mathbb{R}^2$, the allocated task set be $T = \{t_1, t_2, \ldots\}$, and the corresponding coordinates be $\{p_1, p_2, \ldots\}$. Both the robot's position and the task coordinates are mapped into high-dimensional embedding vectors via an embedding function: $e_k = f^e(p_k)$.

To capture spatial dependencies among tasks, an LSTM encoder is employed to extract global contextual features:

$$h_k, h_{\text{last}} = \text{LSTM}(e_k) \tag{14}$$

where $h_k \in \mathbb{R}^d$ denotes the spatial hidden feature of task t_k, and h_{last} is the final hidden state of the LSTM.

Unlike traditional seq2seq models, we do not use the meaningless <start> token as the decoder's initial input. Instead, we replace it with the embedding e_0 of the robot's position:

$$d_0 = \text{decoder}(h_{\text{last}}, e_0) \tag{15}$$

This explicitly incorporates the starting point information, improving the quality of the first task selection.

At each time step of the decoder, the decoder output d_t and the encoder hidden states h_k for all tasks are fed into the attention module. The model then applies a softmax function to compute the selection probability for each task:

$$\pi(t_i \mid T) = \text{Softmax}\left(v^T \cdot \tanh\left(W_1 \cdot h_k + W_2 \cdot d_t\right)\right) \tag{16}$$

where v is the attention weight vector. Here, $\pi(t_i \mid T)$ denotes the probability of selecting task t_i from the allocated task set T. To avoid duplication, a masking operation sets the probability of already selected tasks to zero, ensuring that only the remaining tasks are considered. After each selection, the reward r is defined as the negative distance between the current task and the previously selected task.

5 EXPERIMENT

The experimental evaluation comprises three main parts: ablation study, generalization study, and comparative analysis. All experiments were implemented on a workstation with an Intel(R)

Table 1: Hyperparameters

Name	Description	Value
map_size	map size of the task allocation scenario	20×20
actor_lr	learning rate of actor network	1e-4
critic_lr	learning rate of critic network	1e-4
mix-T_lr	learning rate of self-attention network	1e-4
optimizer	type of optimizer	Adam
α	coefficient of entropy loss	0.05
τ	soft update rate	0.005
batchsize	batch size for global allocation	64
f	activation function	GELU
size	Buffer size for global allocation	1000
γ	discount return	0.98
h	dimensions of all linear hidden layer	128
episode	maximum iteration number	1000

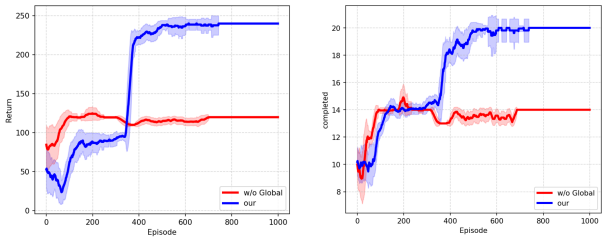
Core(TM) i7-14700KF CPU and an NVIDIA GeForce RTX 4090 GPU. The detailed parameter settings are provided in Table 1.

5.1 Ablation Study

We use the *Individual Selection*, which can independently complete task allocation under the constraints considered in this paper, as the baseline. We evaluate the contributions of the *Global Allocation* and *Sequence Optimization* modules.

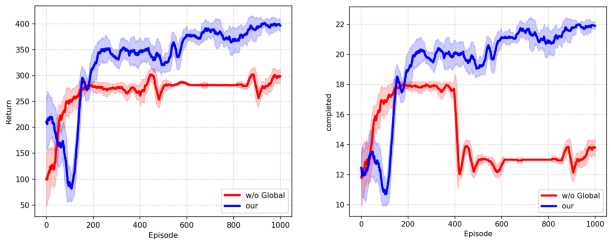
Global Allocation. First, we demonstrate that the global allocation prevents robots from converging to suboptimal solutions. To illustrate this, we construct a small, idealized scenario comprising 5 robots and 20 tasks: 4 specialized robots possess abundant resources for specific types of requirements, while 1 general-purpose robot handles tasks with broadly high demands. Under an optimal allocation, all tasks can be successfully completed. As shown in Fig. 6, when a sequential selection strategy is employed without prior global allocation, the specialized robots tend to greedily select high-demand tasks outside their areas of expertise, rapidly depleting other resources and leaving the remaining tasks unfinished, which leads to resource inefficiency. In contrast, performing global allocation prior to task selection effectively mitigates the risk of local optima, ensures that all tasks are appropriately completed, and achieves the maximum overall return.

To further verify its effectiveness under more typical scenarios, we consider two representative scenarios: (1) **Limited Number of Robots.** Under task-abundant conditions, we deploy 6 robots with the objective of maximizing the overall reward under limited resources. As shown in Fig. 7, the global allocation strategy consistently outperforms the baseline. Notably, when only the Individual Selection method is applied, the number of completed tasks drops sharply after about 400 training epochs. This is due to the lack of global awareness, which makes it difficult for robots to decide whether to pursue high-value tasks or complete a greater number of tasks better suited to their own capabilities. (2) **Limited Number of Tasks.** In this scenario, robots are introduced progressively until all 50 tasks are completed. Every 30 time steps, 2–3 robots are



(a) Overall return (b) Number of completed tasks

Figure 6: Performance comparison in a small-scale idealized scenario. The global allocation significantly improves both overall return and task completion.



(a) Overall return (b) Number of completed tasks

Figure 7: Performance comparison in a typical scenario with constrained robot count.

added to the standby queue. As shown in Fig. 8, an effective allocation strategy can reduce the number of required robots, thereby lowering overall cost and improving the total reward.

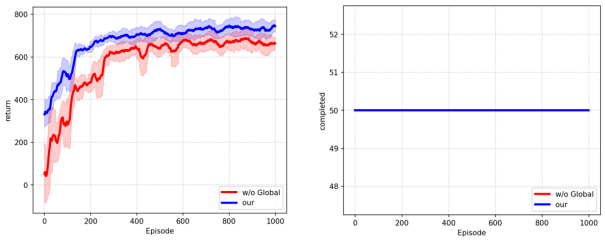
Sequence Optimization. Then, we verify that the Sequence Optimization module can effectively reduce the travel cost. Under the same configuration with the Global Allocation module enabled, we compare two approaches: (i) directly executing the tasks assigned by the upper layer, and (ii) optimizing the execution order after task allocation. As shown in Fig. 9, when only the number of completed tasks and immediate rewards are considered, the two methods perform similarly. However, when the final return is evaluated (i.e., after subtracting each robot's travel cost), the method with Sequence Optimization achieves substantially shorter travel distances. By reordering tasks according to their spatial distribution, it effectively minimizes path costs and yields a markedly higher overall return compared to the unoptimized approach.

5.2 Generalization Study

In the generalization experiments, we first train the model in one scenario and then evaluate its generalization capability by modifying the task environment in three ways: replacing the task set, replacing the robots, and replacing both. For the new task set, each task's resource demand and position are adjusted; for new robots, their appearance frequency and carried resources are modified. We consider three experimental settings: training from scratch, few-shot generalization, and zero-shot generalization.

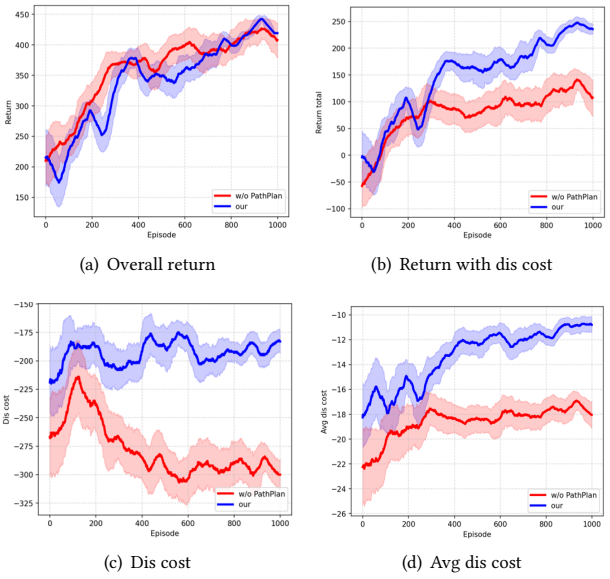
Table 2: Comparison with Other Approaches. The metric Return measures the total reward achieved through task allocation, whereas Dis Cost quantifies the travel cost during task execution. The final Return is derived as the net reward after deducting the corresponding Dis Cost.

Env	metric	Our	SAC	GA	PSO	Auction	Two-Stage
static	total return	263.5 ± 10.2	182.3 ± 20.3	224.2 ± 0.2	261	212	N/A
	return	465.6 ± 11.6	357.0 ± 19.2	412.4 ± 0.2	427	399	N/A
	dis cost	202.1 ± 10.7	174.7 ± 18.5	188.2 ± 0.2	166	187	N/A
dynamic	total return	484.7 ± 40.6	N/A	N/A	N/A	417	344.2 ± 34.6
	return	801.2 ± 36.1	N/A	N/A	N/A	722	783.6 ± 37.1
	dis cost	316.5 ± 39.2	N/A	N/A	N/A	305	439.4 ± 38.2



(a) Overall return (b) Number of completed tasks

Figure 8: Performance comparison in a typical scenario with constrained task count.



(a) Overall return (b) Return with dis cost

(c) Dis cost (d) Avg dis cost

Figure 9: (a) Return without considering movement cost; (b) Final return; (c) Total movement cost; (d) Average movement cost per executed task. It is noteworthy that even without Sequence Optimization, the average distance cost continues to decrease, as task allocation increases the density of executable tasks.

As shown in Fig. 10, across all three modified scenarios, the zero-shot model already achieves a relatively high performance—approximately 80% of that obtained by training from scratch for 1000 epochs. In the few-shot setting, when only tasks or robots are replaced, the pre-trained model reaches the same performance as 1000-epoch from-scratch training after only 200 epochs in the new environment. Even when both tasks and robots are replaced, only 300 epochs are required to achieve comparable performance.

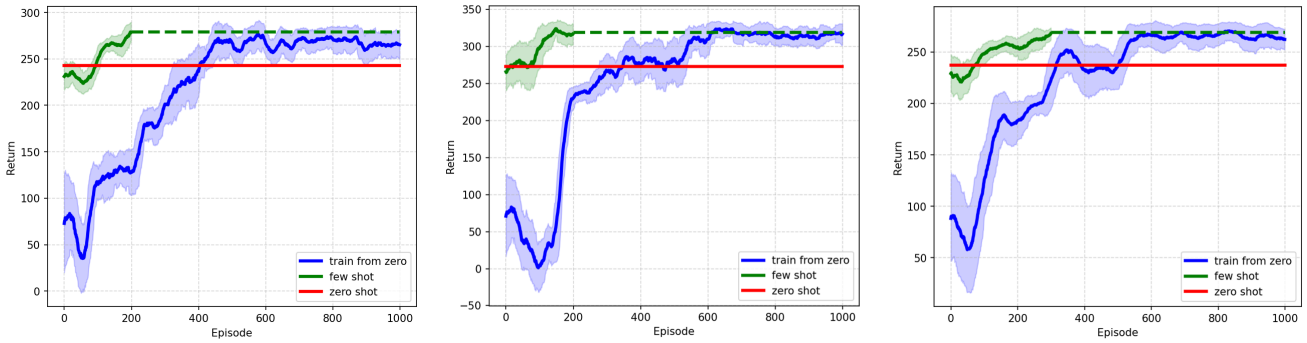
This strong generalization capability stems from the attention mechanism employed in our model, which enables it to learn allocation strategies based on the intrinsic attributes and spatial relationships of tasks and robots, rather than relying on environment-specific state transitions. At the upper-layer task allocation stage, since matching depends on attribute similarity, the model can compute matching probabilities for previously unseen robots through the optimized embedding module. At the lower-layer execution path optimization stage, the pre-trained feature extractor (encoder) and sequential decision module (decoder + attention) can effectively handle unseen TSP instances. Therefore, our model demonstrates strong generalization capability by learning the underlying attribute-matching principles between robots and tasks, rather than overfitting to a specific environment.

5.3 Comparison

Based on the capabilities of existing approaches, we evaluate our method in two distinct environments. **Env. A** features a fixed number of robots, where all agents are introduced at the beginning of the scenario. **Env. B** involves sequentially entering robots, requiring the algorithm to allocate tasks according to scenario-generated trigger conditions at each step. In Environment a, we compare our method with Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Soft Actor-Critic (SAC), and the Auction Algorithm. In Environment b, comparisons are made with the Auction Algorithm and the Two-Stage Reinforcement Learning Algorithm [11]. The detailed configurations of each baseline algorithm are provided in the following.

(1) **Genetic Algorithm (GA)**: Uses integer encoding, with the chromosome length equal to the number of tasks. Each gene value represents the robot assigned to the corresponding task. The parameter settings are as follows: maximum generations = 1000, population size = 50, mutation probability = 0.05, crossover probability = 0.7, and elite retention ratio = 0.1.

(2) **Particle Swarm Optimization (PSO)**: Each particle is represented as an array of length equal to the number of tasks.



(a) tasks replacement (b) robots replacement (c) both replacement

Figure 10: Generalization Performance of Allocation Strategies under Task, Robot, and Combined Replacements

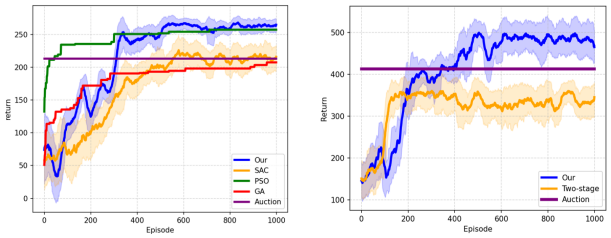
Each element specifies the assigned robot (0: unallocated; 1–N: N-th robot). Particles are decoded into concrete allocation schemes. Parameters: maximum iterations: 1000; swarm size: 50; inertia weight (w): 0.7; cognitive factor (c_1): 1.5; social factor (c_2): 1.5.

(3) **Soft Actor-Critic (SAC)**: Employs the same basic reinforcement learning model as ours. The state vector integrates robot resources, positions, task requirements, locations, availability, and remaining resources. The action space has dimension $M + 1$, representing allocation to a specific robot or no allocation. Basic parameters match our method.

(4) **Auction Algorithm**: Robots act as bidders, with bids defined as net profit from task execution. Tasks are ranked by reward and auctioned sequentially. Each task is allocated to the robot offering the highest marginal profit.

(5) **Two-stage Reinforcement Learning**: Integrates coordinates into robot and task attributes, combining intrinsic and spatial information. During individual selection, tasks are chosen sequentially using a seq2seq-like approach.

As shown in Fig. **??**, even in the relatively static setting of **Environment a**, the proposed method outperforms both the auction-based approach and standard SAC. While our method converges more slowly than heuristic algorithms initially—as the network must learn to extract relevant features from high-dimensional state representations encompassing task requirements, robot capabilities, and spatial relationships—it eventually acquires a policy that efficiently produces high-quality allocations. By the end of training, our approach surpasses GA and matches the performance of PSO. In **Environment b**, the advantage of our method becomes more pronounced. The auction algorithm often yields suboptimal allocations due to its myopic greediness—robots bid only on immediately profitable tasks, ignoring long-term efficiency. The two-stage reinforcement learning approach, while incorporating both positional and capability attributes, fails to model the fundamental difference between nonlinear spatial relationships and directional attribute alignment. This conflation leads to conflicting optimization goals: maximizing resource matching while minimizing travel cost. Our approach resolves these issues via a global matching mechanism



(a) static (b) dynamic

Figure 11: Comparison of our method with other approaches in Environment A and Environment B, where "return" represents the final reward.

and a structured decision process that separates task selection from execution. Comprehensive results are provided in Table 2.

6 CONCLUSION

This paper proposes a three-layer attention-based reinforcement learning framework for dynamic multi-robot task allocation under multiple resource constraints, where robot/task sets vary over time. We hierarchically decompose the problem into task–robot matching and execution sequence optimization via three modules: Global Allocation, Individual Selection, and Sequence Optimization. By dynamically encoding heterogeneous inputs, the framework adapts to varying team/task sizes and generalizes to unseen tasks or robots through learned resource-matching and spatial reasoning. Experiments validate strong performance in both zero-shot and few-shot settings.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their insightful comments on the paper.This work was supported by the National Natural Science Foundation of China (62372415) , Henan Provincial Science Fund for Outstanding Youth (242300421050) and Henan Provincial University Science and Technology Innovation Team (26IRTSTHN034).

REFERENCES

[1] Aakriti Agrawal, Amrit Singh Bedi, and Dinesh Manocha. 2023. RTAW: An Attention Inspired Reinforcement Learning Method for Multi-Robot Task Allocation in Warehouse Environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 1393–1399. https://doi.org/10.1109/ICRA48891.2023.10161310

[2] Muhammad Usman Arif and Sajjad Haider. 2017. An Evolutionary Traveling Salesman Approach for Multi-Robot Task Allocation. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*. INSTICC, SciTePress, 567–574. https://doi.org/10.5220/0006197305670574

[3] Vitor Barbosa, Janine Kniess, and Rafael Parpinelli. 2023. Optimization of Task Allocation in Edge Computing to Industrial Internet with Simulated Annealing. In *Anais do XX Encontro Nacional de Inteligência Artificial e Computacional* (Belo Horizonte/MG). SBC, Porto Alegre, RS, Brasil, 1171–1185. https://doi.org/10.5753/eniac.2023.234651

[4] Xinye Chen, Ping Zhang, Guanglong Du, and Fang Li. 2019-08. A distributed method for dynamic multi-robot task allocation problems with critical time constraints. *Robotics and Autonomous Systems* 118 (Aug 2019-08), 31–46. https://doi.org/10.1016/j.robot.2019.04.012

[5] Han-Lim Choi, Luc Brunet, and Jonathan P. How. 2009. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics* 25, 4 (2009), 912–926. https://doi.org/10.1109/TRO.2009.2022423

[6] Petros Christodoulou. 2019. Soft Actor-Critic for Discrete Action Settings. *CoRR* abs/1910.07207 (2019). arXiv:1910.07207 http://arxiv.org/abs/1910.07207

[7] Katie Clinch, Tony A. Wood, and Chris Manzie. 2023-12. Auction algorithm sensitivity for multi-robot task allocation. *Automatica* 158 (Dec 2023-12), 111239. https://doi.org/10.1016/j.automatica.2023.111239

[8] Fuqin Deng, Huanzhao Huang, Lanhui Fu, Hongwei Yue, Jianmin Zhang, Zexiao Wu, and Tin Lun Lam. 2022. A Learning Approach to Multi-robot Task Allocation with Priority Constraints and Uncertainty. *2022 IEEE International Conference on Industrial Technology (ICIT)* (2022), 1–8. https://api.semanticscholar.org/CorpusID:255478474

[9] Ahmed Elfakharany and Zool Hilmi Ismail. 2021-03. End-to-End Deep Reinforcement Learning for Decentralized Task Allocation and Navigation for a Multi-Robot System. *Applied Sciences* 11, 7 (Mar 2021-03), 2895. https://doi.org/10.3390/app11072895

[10] Haniyeh Fallah, Farzad Didehvar, and Farhad Rahmati. 2020-08. Approximation Algorithms for the Load-Balanced Capacitated Vehicle Routing Problem. *Bulletin of the Iranian Mathematical Society* 47, 4 (Aug 2020-08), 1261–1288. https://doi.org/10.1007/s41980-020-00440-3

[11] Aicheng Gong, Kai Yang, Jiafei Lyu, and Xiu Li. 2024. A two-stage reinforcement learning-based approach for multi-entity task allocation. *Engineering Applications of Artificial Intelligence* 136 (2024), 108906. https://doi.org/10.1016/j.engappai.2024.108906

[12] Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *ArXiv* abs/1801.01290 (2018). https://api.semanticscholar.org/CorpusID:28202810

[13] Ahmed Hussein and Alaa Khamis. 2013. Market-based approach to Multi-robot Task Allocation. In *2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR)*. 69–74. https://doi.org/10.1109/ICBR.2013.6729278

[14] Athira K A, Divya Udayan J, and Umashankar Subramaniam. 2024. A Systematic Literature Review on Multi-Robot Task Allocation. *ACM Comput. Surv.* 57, 3, Article 68 (Nov. 2024), 28 pages. https://doi.org/10.1145/3700591

[15] Xiaodong Li, Yangfei Lin, Zhaoyang Du, Rui Yin, and Celimuge Wu. 2024. Efficient Multi-Robot Cooperative Transportation Scheduling System. In *2024 International Conference on Ubiquitous Communication (Ucom)*. 449–454. https://doi.org/10.1109/Ucom62433.2024.10695882

[16] Song Ma, Jingqing Ruan, Yali Du, Richard Bucknall, and Yuanchang Liu. 2025. An End-to-End Deep Reinforcement Learning Based Modular Task Allocation Framework for Autonomous Mobile Systems. *IEEE Transactions on Automation Science and Engineering* 22 (2025), 1519–1533. https://doi.org/10.1109/TASE.2024.3367237

[17] Xiao Mao, Guohua Wu, Mingfeng Fan, Zhiguang Cao, and Witold Pedrycz. 2025. DL-DRL: A Double-Level Deep Reinforcement Learning Approach for Large-Scale Task Scheduling of Multi-UAV. *IEEE Transactions on Automation Science and Engineering* 22 (2025), 1028–1044. https://doi.org/10.1109/tase.2024.3358894

[18] Brian P. Gerkey;Maja J. Matari pan;. 2004. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research* 23, 9 (2004), 939–954. https://doi.org/10.1177/0278364904045564

[19] Bumjin Park, Cheongwoong Kang, and Jaesik Choi. 2022. Cooperative Multi-Robot Task Allocation with Reinforcement Learning. *Applied Sciences* 12, 1 (2022). https://doi.org/10.3390/app12010272

[20] Ruchir Patel, Eliot Rudnick-Cohen, Shapour Azarm, Michael Otte, Huan Xu, and Jeffrey W. Herrmann. 2020. Decentralized Task Allocation in Multi-Agent Systems Using a Decentralized Genetic Algorithm. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 3770–3776. https://doi.org/10.1109/ICRA40945.2020.9197314

[21] Vyacheslav Petrenko, Fariza Tebueva, Vladimir Antonov, Sergey Ryabtsev, Andrey Pavlov, and Artur Sakolchik. 2023-06. Method and algorithm for task allocation in a heterogeneous group of UAVs in a clustered field of targets. *Journal of King Saud University - Computer and Information Sciences* 35, 6 (Jun 2023-06), 101580. https://doi.org/10.1016/j.jksuci.2023.101580

[22] Han Qingtian. 2021. An Application of Improved PSO Algorithm in Cooperative Search Task Allocation. In *2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA)*. 580–583. https://doi.org/10.1109/ICPECA51329.2021.9362515

[23] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *CoRR* abs/1803.11485 (2018). arXiv:1803.11485 http://arxiv.org/abs/1803.11485

[24] Yan-Tao Tian, Mao Yang, Xin-Yue Qi, and Yong-Ming Yang. 2009. Multi-robot task allocation for fire-disaster response based on reinforcement learning. In *2009 International Conference on Machine Learning and Cybernetics*, Vol. 4. 2312–2317. https://doi.org/10.1109/ICMLC.2009.5212216

[25] Khoa H. Truong, Perumal Nallagownden, Irraivan Elamvazuthi, and Dieu N. Vo. 2020. A Quasi-Oppositional-Chaotic Symbiotic Organisms Search algorithm for optimal allocation of DG in radial distribution networks. *Applied Soft Computing* 88 (2020), 106067. https://doi.org/10.1016/j.asoc.2020.106067

[26] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. *ArXiv* abs/1506.03134 (2015). https://api.semanticscholar.org/CorpusID:5692837

[27] Qi Wang and Chunlei Tang. 2021-12. Deep reinforcement learning for transportation network combinatorial optimization: A survey. *Knowledge-Based Systems* 233 (Dec 2021-12), 107526. https://doi.org/10.1016/j.knosys.2021.107526

[28] Shu Xu, Qingjie Liu, Chengye Gong, and Xupeng Wen. 2025-05. Energy-Efficient Multi-Agent Deep Reinforcement Learning Task Offloading and Resource Allocation for UAV Edge Computing. *Sensors* 25, 11 (May 2025-05), 3403. https://doi.org/10.3390/s25113403

[29] Ruiping Yuan, Jiangtao Dou, Juntao Li, Wei Wang, and Yingfan Jiang. 2023. Multi-robot task allocation in e-commerce RMFS based on deep reinforcement learning. *Mathematical Biosciences and Engineering* 20, 2 (2023), 1903–1918. https://doi.org/10.3934/mbe.2023087

[30] Yi Zhang, Tian Lan, Chuandong Li, Weijie Cai, and Zhiyu Lin. 2025-10. A holomorphic embedding power flow method based on dynamic power restart. *International Journal of Electrical Power & Energy Systems* 171 (Oct 2025-10), 110923. https://doi.org/10.1016/j.ijepes.2025.110923

[31] Xuan Zhou, Xiang Shi, Lele Zhang, Chen Chen, Hongbo Li, Linkang Ma, Fang Deng, and Jie Chen. 2024. Scalable Hierarchical Reinforcement Learning for Hyper Scale Multi-Robot Task Planning. *ArXiv* abs/2412.19538 (2024). https://api.semanticscholar.org/CorpusID:275119206

[32] Xing Zhou, Huaimin Wang, Bo Ding, Tianjiang Hu, and Suning Shang. 2019. Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. *Expert Systems with Applications* 116 (2019), 10–20. https://doi.org/10.1016/j.eswa.2018.09.001

[33] Zilong Zhuang, Zizhao Huang, Yanning Sun, and Wei Qin. 2020-10. Optimization for cooperative task planning of heterogeneous multi-robot systems in an order picking warehouse. *Engineering Optimization* 53, 10 (Oct 2020-10), 1715–1732. https://doi.org/10.1080/0305215x.2020.1821198