Efficient Task Allocation in Multi-agent Systems by Reinforcement Learning and Genetic Algorithm

Zheng Fang , Tao Ma * , Huang Huang , Zhao Niu , Fang Yang

Posted Date: 4 February 2025

doi: 10.20944/preprints202502.0192.v1

Keywords: multi-agent systems; task allocation; reinforcement learning; genetic algorithm

Article

Efficient Task Allocation in Multi-agent Systems by Reinforcement Learning and Genetic Algorithm

Zheng Fang †, Tao Ma *,†, Jun Huang, Zhao Niu and Fang Yang

The College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China; fangzhen@nudt.edu.cn; matao@nudt.edu.cn; huangjun@nudt.edu.cn; niuzhao@nudt.edu.cn; yangfang@nudt.edu.cn

* Correspondence: matao@nudt.edu.cn

† Current address: The College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China.

Abstract: The multi-agent task allocation problem has attracted substantial research interest due to the increasing demand for effective solutions in large-scale and dynamic environments. Despite advancements, traditional algorithms often fall short in optimizing efficiency and adaptability within complex scenarios. To address this shortcoming, we propose a genetic algorithm-enhanced PPO (GAPPO) algorithm, specifically developed to enhance decision-making in challenging task allocation contexts. GAPPO employs a deep reinforcement learning framework, enabling each agent to independently evaluate its surroundings, manage energy resources, and adaptively adjust task allocations in response to evolving conditions. Through iterative refinement, GAPPO achieves balanced task distribution and minimizes energy consumption across diverse configurations. Comprehensive simulations demonstrate that GAPPO consistently outperforms traditional algorithms, resulting in reduced task completion time and heightened energy efficiency. Those findings underscore GAPPO's potential as a robust solution for real-time multi-agent task allocation.

Keywords: multi-agent systems; task allocation; reinforcement learning; genetic algorithm

1. Introduction

The multi-agent task allocation problem involves distributing tasks among multiple agents to optimize resource utilization and minimize operational costs [1], [2]. Effective coordination among agents is essential to enhance performance across a range of applications [3], [4]. Effective task allocation is crucial in those contexts, as it directly impacts the system's overall performance, efficiency, and success [5]. The complexity of this problem lies in balancing task requirements, agent capabilities, and environmental constraints, each of them can vary significantly [6]. Developing robust allocation scheme is crucial to avoid resource wastage and to ensure tasks are assigned to the most suitable agents. As the scale of agents and tasks increases, the computational and logistical demands rise correspondingly, emphasizing the need for adaptive and efficient solutions [7]. In recent years, multi-agent systems have shown significant potential in applications requiring rapid deployment and dynamic task allocation, such as agriculture [8], [9], intelligent transportation [10], [11], and disaster response—including rescue[12], search[13], tracking operations [14], [15], etc. [16].

In earlier years, centralized optimization algorithms, primarily built on models such as mixed-integer programming or reinforcement learning, were extensively employed to address complex agent-assisted network scenarios by utilizing global information from a central controller [17], [18]. For instance, Shabanighazikelayeh *et al*. shown that centralized algorithms can effectively determine optimal UAV placement under high-altitude constraints [19], while Consul *et al*. applied a hybrid federated reinforcement learning framework in a UAV-assisted MEC network to improve computation offloading and resource allocation performance [20]. However, as system scale grows, centralized algorithms reveal limitations. Wang *et al*. [21] emphasized the requirement for adaptability in dynamic environments, it is noting that when centralized algorithms support effective path planning, they struggle with frequent system changes. Al-Hussaini *et al*. [22] proposed an automated task reallocation

 doi:10.20944/preprints202502.0192.v1

method to support human supervisors in managing contingencies during multi-robot missions. However, centralized algorithms face bottlenecks and single points of failure that reduce their adaptability in large-scale, dynamic environments [23]. Those limitations prompt researchers to seek solutions better suited for future networks and complex task allocations.

Distributed optimization algorithms tackle the multi-agent task allocation problem by decentralizing decisions across agents, enabling task allocation based on local information and limited communication with nearby agents. This algorithm enhances scalability and adaptability in dynamic environments, allowing agents to independently respond to system changes without global communication, thereby reducing single points of failure [24]. However, due to the absence of global coordination, distributed algorithms may produce suboptimal allocation and slower convergence, as agents' decisions are constrained by local perspectives, which can lead to coordination challenges [25].

Reinforcement learning (RL) has emerged as a prominent algorithm for addressing the multi-agent task allocation problem [26], [27]. This algorithm enables agents to learn from their interactions within dynamic environments, facilitating the adaptive optimization of task distribution strategies based on real-time feedback [28], [29]. The advantages of RL include its capability to navigate complex scenarios and enhance decision-making processes, which contribute to its effectiveness in multi-agent applications [30], [31]. For instance, Wu *et al*. proposed a triple-dependent multi-agent deep reinforcement learning algorithm for the formation control of agents, incorporating an attention mechanism and adaptive accuracy to enhance agent interactions, coordination, and precision[5]. Ning *et al*. developed a joint optimization algorithm for data acquisition and trajectory planning in UAV-assisted IoT networks, maximizing energy efficiency while adhering to mobility, safety, and task constraints. Their algorithm, framed as a constrained Markov decision process, uses multi-agent deep RL to optimize agent movement policies [32]. Xu *et al*. proposed a federated deep RL algorithm for agent deployment and resource allocation in 6G networks, enabling agents to make real-time decisions based on local observations while pursuing a global optimal solution [33]. Simulation results show that their algorithm improves network throughput and convergence. In addition, Dai *et al*. [34] developed a multi-agent collaborative RL algorithm for agent resource allocation, optimizing interference management and network capacity. Their framework also integrates federated learning to enhance data sharing and cooperation among agents, outperforming traditional RL algorithms in simulation.

Inspired by the potential advantages of integrating different algorithms, we propose a novel algorithm that combines genetic algorithm (GA) with proximal policy optimization (PPO) to effectively address the multi-agent task allocation problem. GA provides robust exploratory capabilities through selection, crossover, and mutation, enhancing the diversity of initial policy generation and improving the search effectiveness within the solution space. In contrast, the PPO component optimizes the learning process through real-time feedback, enabling each agent agent to collect experience by selecting actions and observing rewards, thus dynamically adjusting task allocation. By combining the exploratory strengths of GA with the optimization efficiency of PPO, our algorithm significantly enhances adaptability and effectiveness in tackling the challenges of multi-agent task allocation.

The main contributions of this paper are as follows.

1. We model the multi-agent task allocation problem as a Markov game, where each agent functions as an independent agent, interacting continuously with its environment to iteratively improve its task allocation. Under this framework, each agent determines its actions based on local observations, facilitating coordinated and efficient task distribution across the swarm.
2. We propose a GA-PPO reinforcement learning algorithm, which combines genetic algorithms with proximal policy optimization, incorporating an attention mechanism and adaptive learning rate. This algorithm empowers each agent to autonomously adapt to changing task demands, improving learning efficiency and facilitating effective coordination with multi-agent environments.

 doi:10.20944/preprints202502.0192.v1

By leveraging those features, the system optimizes resource utilization while maintaining robust performance in dynamic scenarios.

3. Numerical experiments demonstrate that our proposed algorithm performs well in terms of convergence speed and scalability compared with the existing representative algorithms.

This paper is organized as follows. Section II describes the problem formulation. Section III presents the GAPPO algorithm and provides its detailed process. Section IV discusses the performance of the algorithm in different environments. Section V presents the summary and future direction, The main notations of this paper is listed in Table 1.

Table 1. Notations in the paper

Notation	Definitions
$n(t)$	the number of agents at time t
m	the number of tasks
N	the set of $n(t)$ agents
S	the set of m tasks
N_j	the number of agents participating in task j
$\{S\}_i$	the task that agent i chooses
$\{I\}_j$	the set of agents or the group for task j
ω_i	the work ability of agent i
h_j	the workload of task j
T_j	the completion time of task j
T	the max completion time of all tasks
A_i	the strategy set of agent i
A	the strategy space
τ	the discount factor reflecting the rate of task reward change
Rad_i	the communication radius of agent i j
R_j	the reward of task j
E_i	the battery level of the agent i
v_i	the speed of the agent i
θ	the elevation angle of the agent

2. Problem Formulation

In this paper, we address the multi-agent task allocation problem, where agents are assigned to distinct tasks and operate either independently or cooperatively to enhance task efficiency and coverage across the environment.

More specifically, agents have the following two tasks.

1. Primary (Extrinsic) Task: High-quality execution of multiple operational tasks by agents, such as conducting surveillance on designated targets or performing search and rescue missions in critical scenarios. Those tasks necessitate that agents autonomously assess their environments and adapt to dynamic conditions, thereby ensuring effective and efficient performance.
2. Secondary (Intrinsic) Task: Evaluation of task allocation decisions. This task focuses on evaluating the effectiveness of agents' decisions in optimizing resource utilization and enhancing operational efficiency, thereby improving the overall performance of the multi-agent system.

To achieve those objectives, which include the high-quality execution of operational tasks and the evaluation of task allocation strategies, each agent undertakes the following steps: state estimation,

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 February 2025 doi:10.20944/preprints202502.0192.v1

task allocation (for scenarios involving multiple tasks), and strategy, as illustrated in Figure 1 and elaborated upon in the subsequent sections.

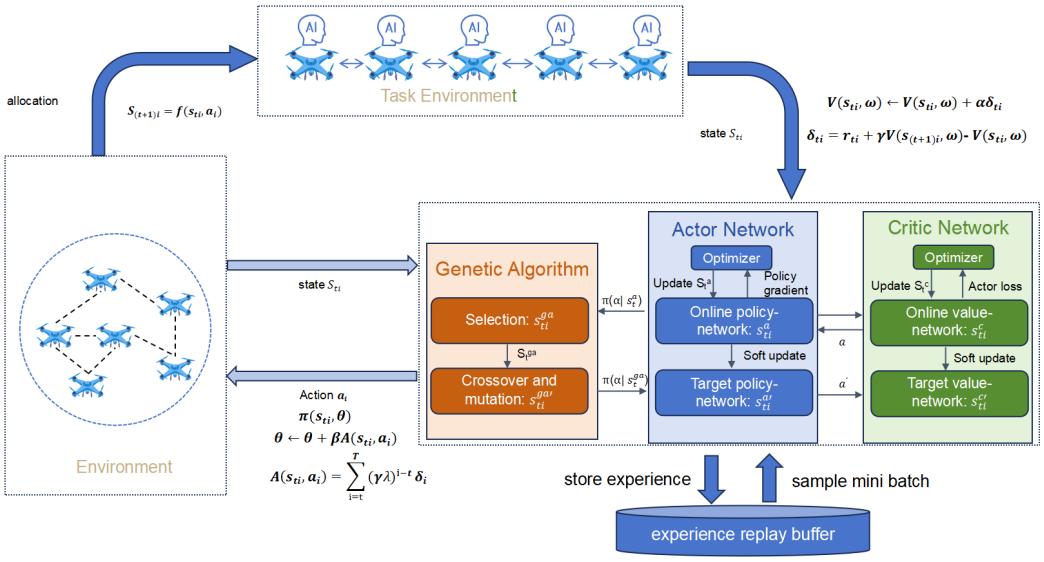

Figure 1. Adaptive task-oriented gappo for agent swarms algorithm.

1. State Estimation: The state information includes the positions of individual agents, an occupancy map of the environment, and the IDs of associated targets. In this multi-agent environment, the state S_{ti} represents the specific configuration of each agent i at time t. Each agent continuously updates its own state information through onboard sensors and communication systems, capturing details about nearby targets and obstacles in the task environment. The task environment models the transition from the current state S_{ti} and action a_i of a agent to the next state $S_{t(i+1)}$ using the function $f(S_{ti}, a_i)$, representing the dynamics of the environment. The comprehensive state representation, including spatial positions, environmental occupancy, and task-related details, is then input into the Actor-Critic network module, which supports policy generation and optimization for intelligent navigation and target detection by each agent.
2. Task Allocation: To optimize task distribution among agents and prevent clustering around the same task, we implement a dynamic task allocation algorithm. Each agent independently evaluates available tasks based on specific criteria, including proximity, urgency, and current workload. The allocation process employs a priority-based strategy, whereby agents rank tasks and assign themselves based on their evaluations. The vital factors influencing this decision include the estimated time required to complete a task and the anticipated benefits of successful execution. In addition, agents communicate detailed task statuses, including progress updates, current allocation status, and priority changes. This information, along with relevant environmental data, is shared with nearby agents, facilitating dynamic task reallocation as conditions change or new tasks emerge. This collaborative algorithm enhances overall task coverage and enables agents to manage resources effectively while adapting to unexpected developments in the operational environment.
3. Strategy: Beginning with the estimated state, each agent must determine its navigation path to optimize joint detection, mapping performance, and overall network efficacy. Initially, each agent functions as an independent learner, estimating its policy and making navigation decisions to improve local outcomes. Upon executing an action, each agent receives an instantaneous reward that reflects the effectiveness of the chosen action, facilitating an initial policy update. In collaborative scenarios, agents can exchange information with neighboring or more experienced agents, such as sharing policy gradients, which contributes to the refinement of their policies. The system employs a GAPPO strategy, allowing agents to make informed navigation decisions autonomously or in coordination with others. GAPPO utilizes attention mechanisms to selec-

 doi:10.20944/preprints202502.0192.v1

tively incorporate shared information, thereby optimizing policy performance and enhancing coordination among agents.

2.1. System Model

Suppose there exist a set of agents formed by $N = \{1, 2, \ldots, n\}$ and a set of tasks formed by $S = \{1, 2, \ldots, m\}$. A agent cannot participate in multiple tasks simultaneously. The task allocation problem in this multi-agent system can be formulated by defining the strategy space for each agent. Let $\{S\}_i$ represent the task selected by agent $i \in N$, and let $\{I\}_j$ represent the set of agents allocated to task $j \in S$. Each task j is positioned at a fixed location $pos_j = (x_j, y_j, z_j)$, defined by its three-dimensional coordinates. The position of agent i, pos_i, coincides with the position of the task it selects. The Euclidean distance between agent i and task j is given by

$$d_{ij} = \|\text{pos}_i - \text{pos}_j\|. \tag{1}$$

Here, d_{ij} also represents a constraint on the communication range of the agents. Specifically, an agent can only access tasks or interact with other agents that are within its communication range, as determined by d_{ij}.

The workload of task j, represented as $h_j \geq 0$, represents the total effort required to complete the task through the combined efforts of the agents. The work ability of agent i is represented by ω_i, indicating the amount of work that agent i can execute per unit time. For task j, the work ability of the group allocated to the task is defined as the sum of the work capacities of all agents in that group $\sum_{i \in I_j} \omega_i$. To complete task j, the total work ability of the allocated group must satisfy

$$\sum\nolimits_{i \in \{I\}_j} \omega_i - h_j \geq 0. \tag{2}$$

For each task $j \in S$, it is crucial to allocate an appropriate number of agents to satisfy (2) and thereby minimize resource wastage due to inefficient allocations. Thus, the reward for task j is expressed as

$$R_j = -\tau^{\sum_{i \in \{I\}_j} \omega_i - h_j}, \tag{3}$$

where discount factor $\tau \in (0, 1)$ represents the rate of change in reward based on the work ability of the group $\{I\}_j$. It also indicates the timeliness of the task. When τ approaches 1, the reward continues to increase even if (2) is satisfied, encouraging more agents to participate and thereby speeding up task completion. Conversely, when τ approaches 0, the reward R_j becomes insensitive to the (2), meaning no additional agents would gain reward from the task once the condition is satisfied.

The objective of the task allocation problem is to identify an optimal allocation that minimizes the total completion time $T = \max\{T_1, T_2, \ldots, T_m\}$, the completion time for task j, which is expressed as

$$T_j = \frac{h_j}{\sum_{i \in \{I\}_j} \omega_i}. \tag{4}$$

Therefore, the problem can be formulated as follows,

$$\begin{aligned} &\min_{\{e_{ij}\}} \{T_j\}, \\ &\text{s.t.} \quad \sum\nolimits_{j=1}^{m} e_{ij} \leq 1, \quad \forall i \in N, \\ &e_{ij} \in \{0, 1\}, \quad \forall i \in N, \forall j \in S, \end{aligned} \tag{5}$$

where e_{ij} is a binary decision variable, if $e_{ij} = 1$ indicates that agent i is allocated to task j; otherwise, agent i is not allocated to task j.

 doi:10.20944/preprints202502.0192.v1

2.2. State Action Model

In our agent swarm framework, each agent's transition at time t is represented by the tuple (s_t, a_t, r_t, s_{t+1}), where:

- s_t denotes the agent's current state, which is a composite representation consisting of:
  - the agent's location $\text{pos}_i = (x_i, y_i, z_i)$, providing the agent's spatial position in the environment. This information is essential for determining the agent's proximity to tasks and other agents, which influences both task allocation and collision avoidance.
  - task-related information, including task priorities, workload status, and the agent's assigned task. This information allows the agent to evaluate its current workload and adjust its decisions based on the urgency and importance of tasks within its scope.
  - the occupancy map, encoding the spatial distribution of tasks and agents within the agent's communication or sensing range. This map helps the agent understand its environment by providing real-time updates on the locations of other agents and tasks, enabling more informed decisions on task allocation and collaboration with other agents.
- a_t is the action selected by the agent in state s_t. The action is defined as the selection of a task $j \in S$, where S represents the set of available tasks. Formally,

$$a_t = \arg\max_{j \in \mathcal{C}(i)} \pi(s_t, j), \tag{6}$$

where $\pi(s_t, j)$ is the policy function that outputs the probability of selecting task j given the current state s_t, and $\mathcal{C}(i)$ is the set of tasks within the communication or sensing range of agent i. If no tasks are feasible, the agent may choose an idle action.
The policy function $\pi(s_t, j)$ is learned using the Actor-Critic architecture, where the Actor is responsible for selecting actions based on the current state, while the Critic evaluates the actions taken by estimating the value function. Specifically, the Actor network takes the current state s_t as input and outputs a probability distribution over possible actions (tasks). The policy function $\pi(s_t, j)$ is expressed as

$$\pi(s_t, j) = P(a_t = j \mid s_t), \tag{7}$$

where a_t is the action selected by the agent at time step t, and j is a task from the set of available tasks $\mathcal{C}(i)$.
The Critic, on the other hand, estimates the expected cumulative reward for the agent starting from state s_t. This is done through the value function $V(s_t)$, which represents the expected long-term reward given the current state. It is formally expressed as

$$V(s_t) = \mathbb{E}\left[\sum_{k=t}^{T} \gamma^{k-t} r_k \mid s_t\right], \tag{8}$$

where r_k is the reward received at time step k, and γ is the discount factor that determines the importance of future rewards.
At each time step, the agent selects an action a_t based on the policy $\pi(s_t, j)$ by maximizing the probability of selecting a task from the set $\mathcal{C}(i)$ based on the current state s_t. The interaction between the Actor and Critic helps in continuously optimizing the policy, with the Critic providing feedback on the expected rewards, guiding the Actor to improve its decision-making over time.
If no feasible tasks are within the agent's sensing or communication range, the agent may choose an idle action, which corresponds to not selecting any task. This idle action is implicitly represented within the action space, where one of the actions is designated as "idle."
- r_t represents the reward obtained after executing action a_t. This reward reflects the agent's contribution to task completion, considering factors such as task importance, completion effectiveness, and collaborative efficiency with other agents.

 doi:10.20944/preprints202502.0192.v1

- s_{t+1} is the subsequent state resulting from the agent's interaction with the environment after performing a_t.

This process encapsulates the agents' decision-making as they interact with the environment, where each agent optimizes its actions to maximize cumulative rewards using a combination of Actor-Critic and genetic algorithms.

In the Proximal Policy Optimization (PPO) framework, the advantage function $A(s_t, a_t)$ evaluates the relative benefit of selecting action a_t in state s_t. PPO employs Generalized Advantage Estimation (GAE) to compute the advantage. The advantage function is given by:

$$A(s_t, a_t) = \hat{A}_t = \sum_{i=t}^{T} (\gamma\lambda)^{i-t} \delta_i, \tag{9}$$

where γ is the discount factor, which controls the importance of future rewards relative to immediate rewards. It is a value between 0 and 1. λ is the smoothing parameter, which helps balance the bias-variance trade-off in estimating the advantage. It is also between 0 and 1.δ_i is the temporal difference error (TD-error), which is expressed as

$$\delta_i = r_i + \gamma V(s_{i+1}; \omega) - V(s_i; \omega), \tag{10}$$

where γ is the discount factor. The value function $V(s_i; \omega)$ is updated iteratively as

$$V(s_i; \omega) \leftarrow V(s_i; \omega) + \alpha\delta_i, \tag{11}$$

with α is the learning rate. This update process ensures that each agent effectively optimizes its cumulative rewards, leading to more precise task execution and resource allocation.The initial value of $V(s_i; \omega)$ is typically set to zero.

Moreover, the GA enhances policy optimization across the agent swarm by selecting and evolving the most effective action strategies. Integrating the GA with the Actor-Critic framework inherent in the PPO algorithm achieves a refined balance between exploration and exploitation. Consequently, this framework significantly improves adaptability in dynamic task environments. The synergy between the GA and the Actor-Critic mechanism of PPO not only fosters effective strategy development but also contributes to the overall robustness and efficiency of the agent swarm in complex operational scenarios.

3. A learning Algorithm

There existing some optimization algorithms for addressing multi-agent task allocation problems, including the leader-follower algorithm[35], the virtual structure algorithm[36], and behavior-based control algorithm[37]. In addition, reinforcement learning has demonstrated effectiveness in this area. To address the challenges associated with dynamic task allocation among agents, this paper presents a novel algorithm, the genetic algorithm-enhanced proximal policy optimization (GA-PPO), which aim is to enhance decision-making and coordination among agents in dynamic environments.

3.1. Genetic Algorithm-Enhanced Proximal Policy Optimization

This algorithm leverages the efficiency of PPO for learning optimal policies while utilizing the evolutionary principles of GA to select the best-performing agents, thereby enhancing decision-making in dynamic environments.

The algorithm commences by initializing a population P of Actor-Critic networks, where each individual in the population represents a distinct policy tailored for agent operations. Key hyper-parameters, including the learning rate lr, discount factor γ, clip ratio ϵ, and KL divergence target KL_{target}, are established to guide the learning process. A replay buffer with a capacity of $C = 10000$ is created to store observed state-action-reward sequences, facilitating experience sampling during training. The generation counter g is initialized to track the evolution of the population.

 doi:10.20944/preprints202502.0192.v1

In each iteration, the algorithm collects trajectories for each individual P_i with the population. Each agent observes the current state s_t and selects an action a_t based on its policy P_i. The selected action is executed, resulting in a reward r_t and a subsequent state s_{t+1}. The transition (s_t, a_t, r_t, s_{t+1}) is appended to the trajectory T, and this process continues until a terminal state is reached. The accumulated trajectories are stored in the replay buffer, allowing the PPO component to sample batches for optimization.

Once the replay buffer contains a sufficient number of samples, the strategy update phase commences. A batch of transitions is sampled, and for each transition, the algorithm computes the returns R_t using the discount factor γ. The advantages $\hat{A}_t$ are calculated utilizing generalized advantage estimation (GAE). The action probabilities $\pi(a_t|s_t)$ and state values $V(s_t)$ are obtained to compute the surrogate loss $L(\theta)$, as previously defined. This surrogate loss plays a crucial role in the GAPPO algorithm by ensuring that policy updates remain stable while allowing for effective exploration. Evaluate P_i in the environment to obtain fitness score F_i using

$$F_i = \alpha \cdot \frac{R_i}{R_{max}} + \beta \cdot \frac{H_i}{H_{max}} - \gamma \cdot \frac{E_i}{E_{max}}, \tag{12}$$

where R_i is the cumulative reward obtained by the individual P_i, H_i is the policy entropy encouraging exploration, and E_i is the computational resource cost. The factors α, β, γ are weights balancing those components.

To ensure the stability of the training process, PPO employs a clipped objective function for policy updates, which is expressed as

$$\begin{aligned} L_1 &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t, \\ L_2 &= \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) \hat{A}_t, \\ L(\theta) &= \min(L_1, L_2), \end{aligned} \tag{13}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ denotes the ratio of the new and old policies, and ϵ controls the permissible range of this ratio. The clip function clip$(\cdot)$ is expressed as

$$\text{clip}(x, 1-\epsilon, 1+\epsilon) = \begin{cases} 1-\epsilon & \text{if } x < 1-\epsilon \\ x & \text{if } 1-\epsilon \leq x \leq 1+\epsilon \\ 1+\epsilon & \text{if } x > 1+\epsilon \end{cases}. \tag{14}$$

This objective function (14) restricts the magnitude of policy updates, thereby preventing significant shifts in the policy and enhancing training stability. Furthermore, incorporating a GA can optimize policy exploration. In each generation, the GA evaluates the fitness of multiple strategies, selecting the best-performing individuals for crossover and mutation, which generates a new set of strategies. This algorithm effectively expands the policy space and enhances the diversity and global optimality of the search.

Additionally, the value loss L_V is calculated as

$$L_V = \frac{1}{2}(R_t - V(s_t))^2. \tag{15}$$

The total loss L is computed by combining the surrogate loss and the value loss, and the network parameters θ is updated through gradient descent.

As the generations progress, our proposed GAPPO algorithm evaluates each individual in the population to obtain fitness scores F_i. Based on those scores, a tournament selection strategy is employed to choose individuals for reproduction. A new population P' is generated through crossover

 doi:10.20944/preprints202502.0192.v1

and mutation, and the original population P is updated accordingly. This evolutionary process continues until the maximum number of generations is reached.

The enhanced PPO agent with genetic algorithm enhances task allocation in multi-agent systems by effectively balancing exploration and exploitation while ensuring diversity within the agent population. This algorithm demonstrates robustness and convergence, even in scenarios with varying numbers of agents. The operational flow of the algorithm is encapsulated in the procedure outlined in Algorithm 1.

Algorithm 1 Pseudo-Code of the GAPPO algorithm

Initialize: Population of Actor-Critic networks $P=\{P_1, \ldots, P_N\}$;
Set hyperparameters: learning rate lr, discount factor γ, clip ratio ϵ, and KL divergence target KL_{target};
Initialize replay buffer with capacity $C = 10000$;
Set generation counter $g = 0$;
while training **do**
  for each individual $P_i \in P$ **do**
    Initialize trajectory T and observe initial state s_0;
    for each time step $t = 0$ to T_{max} **do**
      Select action a_t using policy P_i based on s_t;
      Execute a_t, observe reward r_t and next state s_{t+1};
      Append transition (s_t, a_t, r_t, s_{t+1}) to trajectory T;
      if terminal state reached **then**
        Break;
      end if
    end for
    Store trajectory T in replay buffer;
  end for
  if len(replay buffer) $\geq$ BATCH_SIZE **then**
    Sample a batch of transitions from replay buffer;
    for each transition in the batch **do**
      Compute returns R_t using γ;
      Compute advantages $\hat{A}_t$;
      Get action probabilities $\pi(a_t|s_t)$ and state values $V(s_t)$;
      Calculate surrogate loss $L(\theta)$ by (6);
      Calculate value loss from (7);
      Compute total loss $L = L(\theta) + L_V$;
      Update network parameters θ using gradient descent;
    end for
  end if
end while
while $g <$ max_generations **do**
  for each individual $P_i \in P$ **do**
    Evaluate P_i in the environment to obtain fitness score F_i;
  end for
  Select individuals based on fitness using a selection strategy;
  Generate new population $P' = \{P'_1, \ldots, P'_N\}$;
  Update population $P \leftarrow P'$ and increment generation counter g;
end while
Periodically: Update target networks to stabilize training;

This optimization framework leverages genetic operations—selection, mutation, and recombination—enabling agents to evolve strategies that better adapt to dynamic task environments. By integrating the genetic algorithm with the Actor-Critic network, agents are equipped to refine their strategies through both gradient-based and evolutionary search, enhancing adaptability and ensuring efficient coverage and resource allocation in complex and variable environments.

This section provides a comprehensive overview of the enhanced PPO agent with genetic algorithm, detailing its initialization, iterative interactions between agents and the environment, and the strategy update phase. The distinctive features of this algorithm in the context of multi-agent task allocation are emphasized, highlighting its potential for improving operational efficiency.

4. Experimental Results

In this section, we evaluate the established model alongside the proposed GAPPO algorithm through numerical simulations under varying experimental configurations. Specifically, four experi-

 doi:10.20944/preprints202502.0192.v1

ments were conducted to assess the performance of GAPPO in task allocation, considering scenarios with different numbers of agents and tasks. The results demonstrate that GAPPO consistently outperformed other representative algorithms, achieving efficient task allocations and maintaining robust performance across diverse conditions.

To provide additional insights into the algorithm's effectiveness, a bar chart was employed to compare energy consumption across these scenarios. The findings highlight GAPPO's capability to optimize task allocation while ensuring effective resource utilization. Overall, the evaluation underscores the practicality of GAPPO in addressing complex task allocation problems in multi-agent systems, further validating its potential to meet diverse system requirements.

4.1. Task Allocation

This subsection focuses on evaluating the effectiveness of our proposed algorithm in environment. The GAPPO algorithm is compared with the following representative algorithms.

1. Advantage Actor-Critic (A2C) [38]: The A2C algorithm optimizes decision-making by integrating policy and value functions. The advantage function $A(s_t, a_t)$, is expressed as

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t), \tag{16}$$

   where $Q(s_t, a_t)$ denotes the expected return from action a_t in state s_t, and $V(s_t)$ provides the baseline value of s_t. This structure enhances stability in learning by reducing gradient variance.
2. Proximal Policy Optimization (PPO) [39]: PPO is an on-policy algorithm designed to balance exploration and exploitation by constraining policy updates, which limits excessive updates and enhances stability and sample efficiency in learning.
3. Deep Q-Network (DQN) [40]: DQN is a value-based reinforcement learning algorithm that approximates the optimal action-value function $Q(s, a)$ using deep neural networks. Leveraging experience replay and a target network, DQN stabilizes learning and updates the Q-values as

$$\begin{aligned} \delta &= r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t), \\ Q(s_t, a_t) &\leftarrow Q(s_t, a_t) + \alpha\, \delta, \end{aligned} \tag{17}$$

   where α is the learning rate, r_t is the reward, and γ is the discount factor.
4. Deep Deterministic Policy Gradient (DDPG) [41]: DDPG is an off-policy algorithm that addresses continuous action spaces by combining an actor-critic framework with deterministic policy gradients. The actor network directly optimizes the policy, while the critic network evaluates the action-value function, with the policy gradient update given by

$$\nabla_\theta J \approx \mathbb{E}_{s_t \sim \rho^\pi} \left[\nabla_a Q(s_t, a|\theta^Q) \nabla_\theta \mu(s_t|\theta^\mu) \right], \tag{18}$$

   where $Q(s, a|\theta^Q)$ is the action-value function, $\mu(s_t|\theta^\mu)$ is the policy, and θ^μ and θ^Q are the parameters of the actor and critic networks, respectively.

Four scenarios are examined: i) 100 agents are allocated to 20 tasks; ii) 300 agents are allocated to 30 tasks; iii) 500 agents are allocated to 30 tasks; iv) 500 agents are allocated to 50 tasks.

Monte Carlo simulations are conducted for each scenario, with results generated from over 100 runs per scenario.

In each simulation trial, agents are randomly allocated, with an initial random allocation conducted for each agent at the starting moment, and the maximum number of iterations is set to 200 for the first scenario, while it is set to 300 for the subsequent three scenarios. The time unit Δt is defined as 1 hour (h) across all scenarios.

In the first scenario, the work ability of each agent is randomly allocated an integer value between 1 and 10, ensuring that the total work ability of all agents is exactly 400. Similarly, the workload for

 doi:10.20944/preprints202502.0192.v1

each of the 20 tasks is initialized with a distinct positive value, ensuring that the total workload sums to 400, which maintains balance in the allocation problem. The optimal completion time, as determined through an exhaustive search across all possible allocations, is 1 hour. This serves as the benchmark for assessing the performance of the algorithms under evaluation.

To rigorously compare the effectiveness of different reinforcement learning algorithms, we evaluate the performance of five distinct algorithms—GAPPO, A2C, PPO, DQN, and DDPG—within an environment that simulates a multi-agent system consisting of 100 agents. Each agent is tasked with completing 20 assignments, and the goal is to allocate resources efficiently to minimize the total completion time. The results, as depicted in Figure 2, clearly demonstrate the superior performance of the GAPPO algorithm. GAPPO's maximum completion time decreases rapidly as the number of training iterations increases, stabilizing at a relatively low value early in the training process. This indicates the algorithm's efficiency in converging to optimal solutions and suggests that it is able to learn the task allocation strategy more effectively and quickly compared to its counterparts. In contrast, the A2C, PPO, DQN, and DDPG algorithms exhibit slower learning curves, requiring significantly more iterations to achieve comparable results. These algorithms also demonstrate noticeable fluctuations in performance at various stages of training, suggesting that their learning processes are less stable and more susceptible to suboptimal allocations. These results highlight GAPPO's rapid learning speed, greater stability, and overall superior performance in addressing task allocation challenges at this scale, making it a promising candidate for large-scale multi-agent systems.

In the second scenario, the problem complexity increases as 300 agents are tasked with completing 30 assignments. The total execution ability of the agents is 1200, which exactly matches the overall task load of 1200, creating a more challenging allocation environment. As in the first scenario, an exhaustive search is performed to determine the optimal completion time, which remains 1 hour, providing a consistent benchmark for comparison.

The results in Figure 3 reveal that GAPPO's advantages become increasingly apparent as the task scale expands. Despite the increased complexity—driven by the larger number of agents and tasks—GAPPO continues to perform remarkably well, achieving substantial performance improvements in fewer iterations. Its learning curve exhibits a consistently smooth and steep downward trajectory, indicating that GAPPO is able to effectively adapt to the increased problem size without experiencing the instability observed in other algorithms. In comparison, the A2C, PPO, DQN, and DDPG algorithms show more erratic performance, with significant fluctuations and slower convergence. These algorithms require more iterations to stabilize their learning process and do not demonstrate the same level of smooth performance improvement as GAPPO. The ability of GAPPO to maintain a smooth, steady reduction in completion time—especially in the later stages of training—emphasizes its superior generalization ability and robustness. This shows that GAPPO is not only faster but also more capable of handling the increased complexity associated with larger task scales. The ability to maintain a stable learning trajectory under these more challenging conditions further demonstrates GAPPO's adaptability and resilience in solving complex task allocation problems.

In the third scenario, the problem complexity continues to escalate as 500 agents are allocated to 30 tasks. The total execution ability of the agents is 2400, which precisely matches the overall task load of 2400, keeping the task allocation problem balanced. As before, the optimal completion time is determined through exhaustive search, and the benchmark remains at 1 hour.

Figure 4 illustrates that GAPPO continues to exhibit remarkable performance even in this large-scale setup. The learning curve of GAPPO decreases rapidly during the initial stages of training, reflecting its ability to quickly converge to effective solutions. More importantly, GAPPO maintains robust performance throughout the entire training period, with no significant fluctuations or signs of stagnation. This stability is crucial in managing large-scale systems, where erratic learning behaviors could lead to suboptimal task allocations and wasted computational resources. In contrast, the other algorithms—including A2C, PPO, DQN, and DDPG—show inconsistent learning curves, with noticeable fluctuations in performance and slower convergence toward the optimal solution.

 doi:10.20944/preprints202502.0192.v1

These irregularities highlight the challenges faced by these algorithms in managing larger-scale systems effectively. GAPPO's ability to maintain steady performance, especially in the later stages of training, reflects its superior scalability, efficiency, and reliability in handling increasingly complex multi-agent environments. The effectiveness of GAPPO in dealing with this large-scale scenario further underscores its potential for real-world applications involving large numbers of agents and tasks.

In the fourth scenario, the complexity of the problem reaches its peak as 500 agents are tasked with completing 50 assignments. The total execution ability of the agents is 2400, which matches the total task load of 2400, continuing to maintain balance between the agents' capacities and the tasks' demands. An exhaustive search determines the optimal completion time to be 1 hour, and this value remains consistent across scenarios, providing a reliable baseline for algorithm evaluation.

The results, shown in Figure 5, highlight the ability of GAPPO to excel even under the most demanding conditions. Despite the exponential increase in task volume and complexity, GAPPO is able to significantly reduce the maximum completion time, achieving optimal performance in a remarkably short period. This demonstrates GAPPO's ability to handle large-scale multi-task allocation problems effectively, with a nearly vertical decline in the learning curve during the initial phases of training. This rapid convergence stands in stark contrast to the other algorithms, whose learning curves are slower and characterized by noticeable rebounds or regressions. GAPPO's ability to maintain such a sharp and consistent reduction in completion time is a testament to its powerful computational capability and efficient decision-making mechanism. The impressive speed at which GAPPO reaches optimal performance further emphasizes its suitability for complex, large-scale task allocation problems, where quick and reliable solutions are essential.

Through a comparative analysis of these four experimental scenarios, it is evident that GAPPO consistently performs well across a broad range of task scales and complexities. Whether the problem involves a smaller or larger number of agents and tasks, GAPPO consistently demonstrates its ability to identify optimal solutions while maintaining high consistency and reliability throughout the training process. This makes GAPPO an ideal choice for addressing task allocation challenges in a wide array of practical applications, from small-scale systems to large, complex environments. The robust performance observed in these scenarios validates GAPPO's exceptional efficiency, scalability, and adaptability, confirming its potential for solving multi-agent reinforcement learning tasks in dynamic and large-scale settings. Given these results, we conclude that GAPPO exhibits outstanding performance in reinforcement learning tasks, particularly those requiring efficiency, precision, and robust scalability.

Moreover, GAPPO's efficiency in managing energy consumption represents a significant advantage in the context of multi-agent task allocation systems. As illustrated in Figure 6, an analysis of energy management across different experimental setups demonstrates that GAPPO consistently outperforms other algorithms in terms of battery retention. In the initial setup, involving 100 agents and 20 tasks, GAPPO exhibited remarkable energy efficiency, maintaining a high battery percentage of 91.9% by the end of the task allocation process. In comparison, other algorithms, such as A2C, PPO, DQN, and DDPG, showed significantly lower battery levels, ranging from 52.2% to 55.5%. This disparity highlights GAPPO's exceptional ability to conserve energy while still achieving optimal task allocation performance. Notably, the observed difference in battery retention is particularly striking given that the operational conditions, including task complexity and agent count, were held constant across these algorithms, further emphasizing GAPPO's superior energy management capabilities.

Each agent's battery level is initialized at 20, and the reduction in battery power is proportional to the distance traveled. Therefore, the battery consumption is represented as:

$$E_i^{k+1} = E_i^k - k \cdot d_{ij}, \tag{19}$$

where k is a proportionality constant representing the battery consumption rate per unit distance.

As we moved to more complex experimental setups, where the agent count was increased to 300 and 500, the initial battery capacity for each agent was also adjusted to accommodate the

 doi:10.20944/preprints202502.0192.v1

larger-scale environments. Specifically, the initial battery capacity was increased from 20 to 30 units, which provided a higher baseline for all algorithms. Despite this enhancement, GAPPO continued to demonstrate a clear edge in energy management. In these larger-scale setups, GAPPO's battery percentage remained consistently high, ranging between 91.7% and 91.8%, while other algorithms showed more moderate battery levels. Specifically, A2C, PPO, DQN, and DDPG algorithms converged to battery levels ranging from 74.2% to 81.6%. This sustained advantage, even with a larger initial battery capacity and increased task load, further reinforces the scalability and robustness of GAPPO's energy optimization. The algorithm's ability to maintain such high battery retention under these conditions suggests that GAPPO's energy management is both efficient and resilient, even when the complexity of the problem increases.

Overall, these findings firmly establish GAPPO as a highly efficient algorithm in terms of energy management across various scales and battery configurations. The ability to consistently optimize battery usage without sacrificing performance, even as task complexity and agent numbers grow, is a particularly valuable trait in dynamic, large-scale multi-agent systems. In environments where efficient energy utilization is paramount to sustaining operations over time, GAPPO's energy conservation features are crucial for ensuring long-term operational efficiency. Moreover, GAPPO's ability to strike a balance between task allocation effectiveness and energy optimization makes it a formidable choice for real-world applications where both computational and energy resources are limited. This robust performance across both task allocation and energy management underscores GAPPO's potential for deployment in complex operational environments, where maintaining high efficiency across multiple dimensions is essential for success.

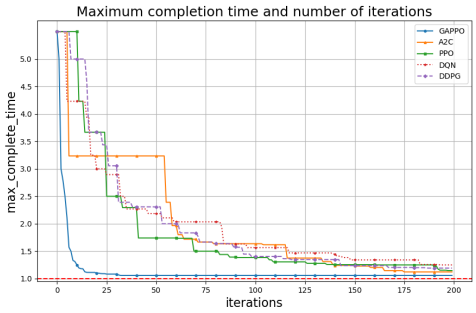

Figure 2. Convergence performance of five algorithms in the environment with 100 agents and 20 tasks. The pink dotted line indicate the value of the optimal solution, which is 1 h.

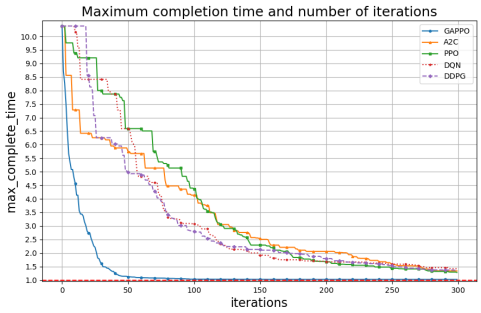

Figure 3. Convergence performance of five algorithms in the environment with 300 agents and 30 tasks. The pink dotted line indicate the value of the optimal solution, which is 1 h.

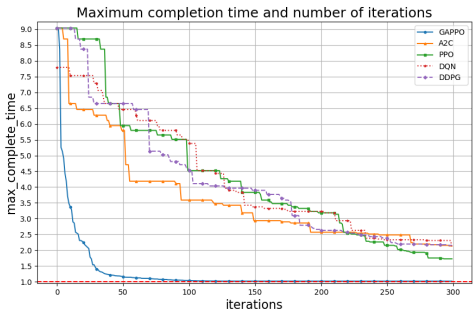

Figure 4. Convergence performance of five algorithms in the environment with 500 agents and 30 tasks. The pink dotted line indicate the value of the optimal solution, which is 1 h.

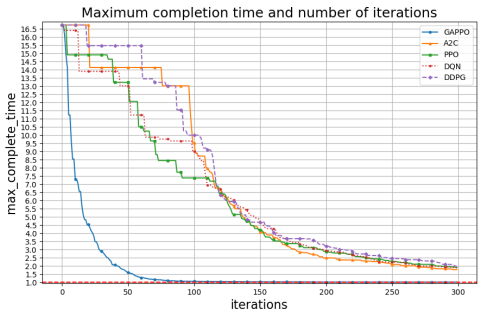

Figure 5. Convergence performance of five algorithms in the environment with 500 agents and 50 tasks. The pink dotted line indicate the value of the optimal solution, which is 1 h.

 doi:10.20944/preprints202502.0192.v1

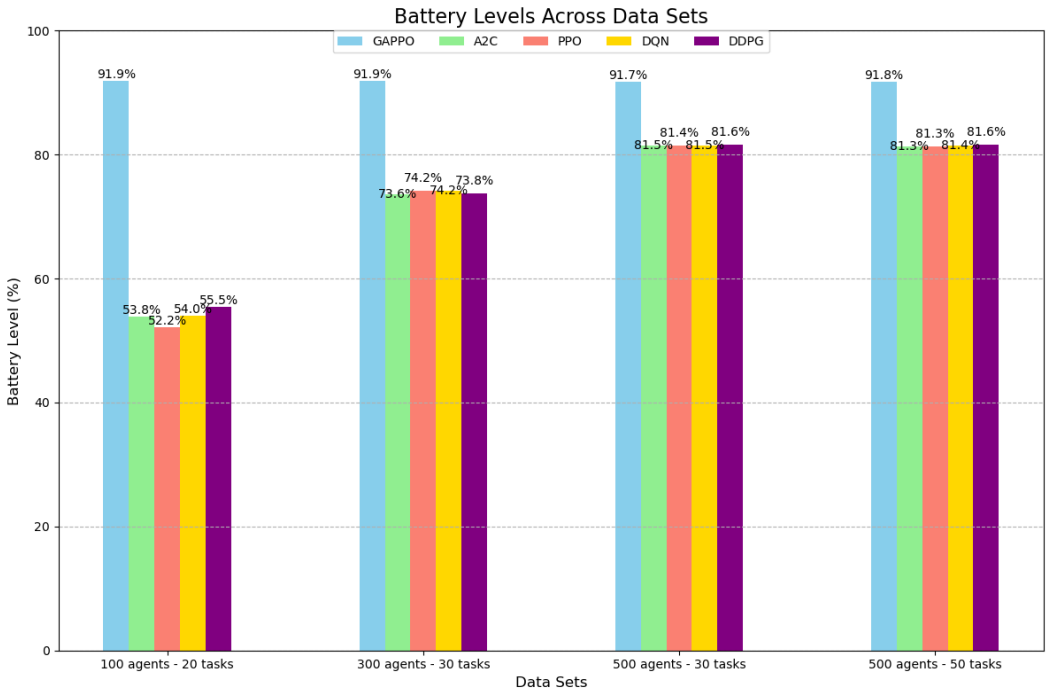

Figure 6. Comparison of Five Algorithms on Power Remaining Percentage After Task Completion.

5. Conclusions and the Future Work

In conclusion, the GAPPO algorithm stands out as a robust solution to the challenges inherent in multi-agent task allocation within complex environments. Traditional algorithms often encounter significant limitations in scalability and adaptability, struggling to manage the intricacies associated with numerous agents and tasks efficiently. In contrast, GAPPO algorithm effectively addresses those shortcoming by prioritizing energy efficiency and optimal resource distribution, leading to superior performance outcomes. Our comprehensive experimental results consistently demonstrate that GAPPO algorithm significantly reduces task completion time while maintaining high energy levels across agent fleets. This effectiveness highlights the algorithm's capability to enhance operational efficiency in diverse scenarios, solidifying its role in advancing multi-agent systems.

Future research will concentrate on several pivotal points: 1) Optimizing energy management strategies within GAPPO, particularly in scenarios with varying task loads and agent configurations; 2) Validating GAPPO's performance in real-world applications to ensure its effectiveness in practical multi-agent systems. In addition, ongoing efforts will explore enhancing GAPPO's adaptability by integrating advanced optimization techniques, ensuring that the algorithm remains at the forefront of multi-agent task allocation methodologies. Those directions aim to strengthen GAPPO's framework and extend its applicability, ultimately contributing to the advancement of efficient and reliable solutions in the field.

References

1. Chen, J.; Guo, Y.; Qiu, Z.; Xin, B.; Jia, Q.-S.; Gui, W. Multiagent dynamic task assignment based on forest fire point model. *IEEE Trans. Autom. Sci. Eng.* 2022, *19*, 833–849.
2. Liu, S.; Feng, B.; Bi, Y.; Yu, D. An Integrated Approach to Precedence-Constrained Multi-Agent Task Assignment and Path Finding for Mobile Robots in Smart Manufacturing. Appl. Sci. **2024**, *14*, 3094.
3. Huang, L.; Wu, Y.; Tempini, N. A Knowledge Flow Empowered Cognitive Framework for Decision Making With Task-Agnostic Data Regulation. *IEEE Trans. Artif. Intell.* 2024, *5*, 2304–2318.
4. Zhuang, H.; Lei, C.; Chen, Y.; Tan, X. Cooperative Decision-Making for Mixed Traffic at an Unsignalized Intersection Based on Multi-Agent Reinforcement Learning. Appl. Sci. **2023**, *13*, 5018.
5. Wu, J.; Li, D.; Yu, Y.; Gao, L.; Wu, J.; Han, G. An attention mechanism and adaptive accuracy triple-dependent MADDPG formation control method for hybrid UAVs. *IEEE Trans. Intell. Transp. Syst.* 2024, *25*, 11648–11663.

 doi:10.20944/preprints202502.0192.v1

6. Yu, Y.; Zhai, Z.; Li, W.; Ma, J. Target-Oriented Multi-Agent Coordination with Hierarchical Reinforcement Learning. Appl. Sci. **2024**, *14*, 7084.
7. Wu, J.; Zhang, J.; Sun, Y.; Li, X.; Gao, L.; Han, G. Multi-UAV collaborative dynamic task allocation method based on ISOM and attention mechanism. *IEEE Trans. Veh. Technol.* 2024, *73*, 6225–6235.
8. Li, W.; Wang, X.; Jin, B.; Luo, D.; Zha, H. Structured Cooperative Reinforcement Learning With Time-Varying Composite Action Space. *IEEE Trans. Pattern Anal. Mach. Intell.* 2022, *44*, 8618–8634.
9. Furch, A.; Lippi, M.; Carpio, R. F.; Gasparri, A. Route optimization in precision agriculture settings: A multi-Steiner TSP formulation. *IEEE Trans. Autom. Sci. Eng.* 2023, *20*, 2551–2568.
10. Fatemidokht, H.; Rafsanjani, M. K.; Gupta, B. B.; Hsu, C.-H. Efficient and secure routing protocol based on artificial intelligence algorithms with UAV-assisted for vehicular ad hoc networks in intelligent transportation systems. *IEEE Trans. Intell. Transp. Syst.* 2021, *22*, 4757–4769.
11. Gong, T.; Zhu, L.; Yu, F. R.; Tang, T. Edge Intelligence in Intelligent Transportation Systems: A Survey. *IEEE Trans. Intell. Transp. Syst.* 2023, *24*, 8919–8944.
12. Ribeiro, R. G.; Cota, L. P.; Euzebio, T. A. M.; et al. Unmanned aerial vehicle routing problem with mobile charging stations for assisting search and rescue missions in post-disaster scenarios. *IEEE Trans. Syst. Man Cybern. Syst.* 2021, *52*, 6682–6696.
13. Wang, H.; Wang, C.; Zhou, K.; Liu, D.; Zhang, X.; Cheng, H. TEBChain: A trusted and efficient blockchain-based data sharing scheme in UAV-assisted IoV for disaster rescue. *IEEE Trans. Netw. Serv. Manag.* 2024, *21*, 4119–4130.
14. Sampedro, C.; Rodriguez-Ramos, A.; Bavle, H.; Carrio, A.; De la Puente, P.; Campoy, P. A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *J. Intell. Robotic Syst.* 2019, *95*, 601–627.
15. Meng, W.; He, Z.; Su, R.; Yadav, P. K.; Teo, R.; Xie, L. Decentralized multi-UAV flight autonomy for moving convoys search and track. *IEEE Trans. Control Syst. Technol.* 2017, *25*, 1480–1487.
16. Liu, Z.; Qiu, C.; Zhang, Z. Sequence-to-Sequence Multi-Agent Reinforcement Learning for Multi-UAV Task Planning in 3D Dynamic Environment. Appl. Sci. **2022**, *12*, 12181.
17. Wang, L.; Zhang, H.; Guo, S.; Yuan, D. Deployment and association of multiple UAVs in UAV-assisted cellular networks with the knowledge of statistical user position. *IEEE Trans. Wireless Commun.* 2022, *21*, 6553–6567.
18. Dai, Z.; Zhang, Y.; Zhang, W.; Luo, X.; He, Z. A multi-agent collaborative environment learning method for UAV deployment and resource allocation. *IEEE Trans. Signal Inf. Process. Netw.* 2022, *8*, 120–130.
19. Shabanighazikelayeh, M.; Koyuncu, E. Optimal placement of UAVs for minimum outage probability. *IEEE Trans. Veh. Technol.* 2022, *71*, 9558–9570.
20. Consul, P.; Budhiraja, I.; Garg, D.; Kumar, N.; Singh, R.; Almogren, A. S. A hybrid task offloading and resource allocation approach for digital twin-empowered UAV-assisted MEC network using federated reinforcement learning for future wireless network. *IEEE Trans. Consum. Electron.* 2024, *70*, 3120–3130.
21. Wang, N.; Liang, X.; Li, Z.; Hou, Y.; Yang, A. PSE-D model-based cooperative path planning for UAV and USV systems in antisubmarine search missions. *IEEE Trans. Aerosp. Electron. Syst.* 2024, *60*, 6224–6240.
22. Al-Hussaini, S.; Gregory, J. M.; Gupta, S. K. Generating Task Reallocation Suggestions to Handle Contingencies in Human-Supervised Multi-Robot Missions. *IEEE Trans. Autom. Sci. Eng.* 2024, *21*, 367–381.
23. Raja, G.; Anbalagan, S.; Ganapathisubramaniyan, A.; Selvakumar, M. S.; Bashir, A. K.; Mumtaz, S. Efficient and secured swarm pattern multi-UAV communication. *IEEE Trans. Veh. Technol.* 2021, *70*, 7050–7058.
24. Liu, D.; Wang, J.; Xu, K. Task-driven relay assignment in distributed UAV communication networks. *IEEE Trans. Veh. Technol.* 2019, *68*, 11003–11017.
25. Chen, Y.; Yang, D.; Yu, J. Multi-UAV task assignment with parameter and time-sensitive uncertainties using modified two-part wolf pack search algorithm. *IEEE Trans. Aerosp. Electron. Syst.* 2018, *54*, 2853–2872.
26. Wu, G.; Liu, Z.; Fan, M.; Wu, K. Joint task offloading and resource allocation in multi-UAV multi-server systems: An attention-based deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* 2024, *73*, 11964–11978.
27. Guo, H.; Wang, Y.; Liu, J.; Liu, C. Multi-UAV cooperative task offloading and resource allocation in 5G advanced and beyond. *IEEE Trans. Wireless Commun.* 2024, *23*, 347–359.
28. Liu, D.; Dou, L.; Zhang, R.; Zhang, X.; Zong, Q. Multi-agent reinforcement learning-based coordinated dynamic task allocation for heterogeneous UAVs. *IEEE Trans. Veh. Technol.* 2023, *72*, 4372–4383.
29. Zhao, N.; Ye, Z.; Pei, Y.; Liang, Y.-C.; Niyato, D. Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing. *IEEE Trans. Wireless Commun.* 2022, *21*, 6949–6960.

 doi:10.20944/preprints202502.0192.v1

30. Mao, X.; Wu, G.; Fan, M.; Cao, Z.; Pedrycz, W. DL-DRL: A double-level deep reinforcement learning approach for large-scale task scheduling of multi-UAV. *IEEE Trans. Autom. Sci. Eng.* 2024, 1–17.
31. Wang, Y.; He, Y.; Yu, F. R.; Lin, Q.; Leung, V. C. M. Efficient resource allocation in multi-UAV assisted vehicular networks with security constraint and attention mechanism. *IEEE Trans. Wireless Commun.* 2023, *22*, 4802–4813.
32. Ning, N.; Ji, H.; Wang, X.; et al. Joint optimization of data acquisition and trajectory planning for UAV-assisted wireless powered Internet of Things. *IEEE Trans. Mobile Comput.* 2024, 1–17.
33. Xu, X.; Feng, G.; Qin, S.; Liu, Y.; Sun, Y. Joint UAV deployment and resource allocation: A personalized federated deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* 2024, *73*, 4005–4018.
34. Dai, Z.; Zhang, Y.; Zhang, W.; Luo, X.; He, Z. A multi-agent collaborative environment learning method for UAV deployment and resource allocation. *IEEE Trans. Signal Inf. Process. Netw.* 2022, *8*, 120–130.
35. Ren, W.; Beard, R. W. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Trans. Autom. Control* 2005, *50*, 655–661.
36. Low, C. B. A dynamic virtual structure formation control for fixed-wing UAVs. *2011 9th IEEE Int. Conf. Control Autom.* 2011, 627–632.
37. Balch, T.; Arkin, R. C. Behavior-based formation control for multi-robot teams. *IEEE Trans. Robotics Autom.* 1998, *14*, 926–939.
38. Konda, V.; Tsitsiklis, J. Actor-critic algorithms. In *Adv. Neural Inf. Process. Syst.* 1999, *12*.
39. Zhang, H.; Jiang, M.; Liu, X.; Wen, X.; Wang, N.; Long, K. PPO-based PDACB traffic control scheme for massive IoV communications. *IEEE Trans. Intell. Transp. Syst.* 2023, *24*, 1116–1125.
40. Mnih, V.; Kavukcuoglu, K.; Silver, D. Human-level control through deep reinforcement learning. *Nature* 2015, *518*, 529–533.
41. Zheng, K.; Jia, X.; Chi, K.; Liu, X. DDPG-based joint time and energy management in ambient backscatter-assisted hybrid underlay CRNs. *IEEE Trans. Commun.* 2023, *71*, 441–456.