2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

TACC: Multi-agent Reinforcement Learning for Task Allocation with Communication Coordination in UAV Swarms

Zehao Xiong, Yexun Xi, Yizhe Cao, Chuan Li, Rong Li, Lixian Shen and Jie Li*

Abstract— **Task allocation in UAV swarms is increasingly challenging due to task complexity, communication limits, and algorithm robustness. Combining reinforcement learning with task allocation offers promise but often ignores the conflict between task conflicts and communication overhead, causing exploration and stability issues. This paper proposes Task Allocation with Communication Coordination (TACC), which learns a gated mechanism to balance communication efficiency and allocation reliability. TACC is modeled as a POMDP with adaptive gating actions and shared rewards for task conflicts, and an asynchronous experience aggregation method is designed for CTDE. We further introduce Multi-Objective Constrained Policy Optimization (MOCPO), which applies constrained policy optimization via a Lagrangian loss to stabilize training and improve convergence. Finally, sim-to-real experiments are conducted in the HIL environment, and the results demonstrate the optimal trade-off achieved by the proposed method and its overall state-of-the-art approaches. Ablation studies and hyperparameter experiments further validated the stability of MOCPO. Specifically, the communication strategy is effectively deployed in the RK3588 SOC, and the flight experiment demonstrates the superior scheduling outcomes of TACC within the ten-UAV swarm in the search and rescue.**

I. INTRODUCTION

Multi-agent task allocation is nothing new, but the swarm boom in recent years has activated it again [1], [2]. As an important application of multi-agent theory, swarms endow multi-agent systems with new characteristics, such as flexibility, locality, and autonomy [3], [4], which makes them highly valued in engineering and presented in various forms, such as the UAV swarm or other agents with individual dynamics. Task allocation is becoming increasingly complex due to the complexity of tasks, the limitations on communication, and the robustness of the allocation algorithm [5], [6]. In addition, Multi-agent task allocation is NP-hard, and a variety of approximate methods have been developed to efficiently produce solutions to this problem [7], [8].

Based on the distribution of task-related computations in multi-agent systems, allocation methods can be categorized into centralized and distributed approaches. Compared to centralized methods, distributed methods have advantages in terms of communication overhead, single-point computation, and global robustness. The distributed method periodically iterates through two stages: computation and communication [9], [10]. In the computation stage, the agent selects appropriate tasks by independently utilizing the sequential greedy algorithm (SGA) to build a task sequence. In the communication stage, agents obtain allocation information maintained by others through communication and then use consensus conflict resolution rules to achieve global agreement [11], [12], [13], [14]. Unfortunately, due to factors such as interdependence among tasks, resource constraints, and dynamic environmental changes, the task allocation problem presents a vast solution space and a complex optimization process, which makes traditional methods unable to conduct an exhaustive search or global optimization of the solution space within a limited time.

Multi-agent reinforcement learning (MARL) can update strategies in real-time based on environmental feedback, with each agent continuously adjusting its behaviour according to local observations to cope with new tasks or changing scenarios, making distributed task allocation more real-time and robust in dynamic and uncertain environments [15]. The sub-phase-based method designs MARL separately for the two-stage process of distributed task allocation, aiming to learn computation or communication strategies. This method has good interpretability due to its consideration of the specific process of task allocation. For the computation stage, a heuristic computational policy is learned to swiftly construct allocations that maximize the global reward [16]. For the communication stage, communication-related strategies, such as transmission loss, protocol parameters, and communication timing, are learned to minimize communication overhead in the task allocation process [17], [18]. Since the distributed task allocation requires consensus among agents, its robustness and timeliness heavily depend on communication, which is often affected by environmental factors such as complex terrain or electromagnetic environment, resulting in various degrees of error, packet loss, delay, and even interruption, which frequently leads to message retransmission, delaying the allocation process. Therefore, communication-related strategy learning holds more promising prospects for distributed task allocation, and it can be independently designed without considering algorithms and protocols compared to computation strategy learning, offering better generalization capabilities and applicability.

Nevertheless, existing research has overlooked the structural conflict between task conflicts and communication overhead, which leads to significant challenges in exploration and training instability. In multi-agent reinforcement learning, Constrained Multi-Agent Reinforcement Learning (C-MARL) guides the direction of policy updates by introducing a constraint mechanism, balancing structural conflicts among multiple optimization objectives during the training process. To this end, this paper proposes a C-MARL method

The authors are with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China lijie09@nudt.edu.cn

with constrained policy optimization that coordinates asynchronous communication timing while balancing multiple objectives, thereby enhancing the effectiveness of task allocation in adapting to realistic communication environments.

II. PRELIMINARIES

A. Distributed Task Allocation Considering Communication Processes

1) Task Allocation Problem Description: To formalize the task allocation problem, let $\mathbf{U} = [u_1, u_2, \ldots, u_n]$ be the set of n agents and $\mathbf{K} = [k_1, k_2, \ldots, k_m]$ be the set of m tasks. Each agent u_j selects candidate tasks from $\mathbf{K}$ to form a task list $\mathbf{p}_j$. The goal of task allocation is to assign tasks to all agents without conflicts to maximize the global reward J:

$$J = \max\{\sum_{j=1}^{n}\sum_{k=1}^{|\mathbf{p}_j|} c_{j,k}(\mathbf{p}_j)\} \tag{1}$$

Where the reward of task k in the task list $\mathbf{p}_j$ is defined as $c_{j,k}(\mathbf{p}_j)$. The goal of task allocation is to maximize the global task reward, which is the sum of rewards for tasks allocated to all agents. The number of tasks allocated to agent u_j (u_j *can be simplified to* j) is $|\mathbf{p}_j|$. Constraints related to the task allocation problem are as follows:

$$\begin{cases} |\mathbf{p}_j| \leq N_j, & \forall j \in \{1,\ldots,n\} \\ \mathbf{p}_j \cap \mathbf{p}_i = \emptyset, & \forall j \neq i \in \{1,\ldots,n\} \\ \bigcup_j^z \mathbf{p}_j \subseteq \mathbf{K}, & \forall j,z \in \{1,\ldots,n\} \end{cases} \tag{2}$$

Where the first constraint ensures that the maximum capacity of agent u_j is N_j. The second constraint ensures that each task is allocated to at most one agent, or not allocated at all, to avoid exceeding the capacity of any agent. The validity of the allocation is guaranteed by the third constraint, which states that any combination of allocations must be a subset of the task set $\mathbf{K}$.

2) Two-Stage for the Distributed Method: Compared to synchronous task allocation, asynchronous methods have more advantages in practical engineering applications as they reduce extra data transmission and waiting time. This paper studies a process based on an asynchronous task allocation algorithm, where agents iteratively undergo two phases: computation and communication[19]. During the computation stage, each agent independently runs a distributed task allocation algorithm to select appropriate tasks while storing task allocation information received from other agents. Once the received information is updated, an agent will move directly to the next stage without waiting for others. In the communication stage, agents use the obtained information and consistent conflict resolution rules to ensure that the allocation results among the nearest agents remain consistent.

III. APPROACH

A. Multi-Agent Reinforcement Learning Environment for Task Allocation with Communication Coordination

As discussed earlier, the effectiveness of distributed task allocation is closely tied to the underlying communication process [20], [21]. By learning communication strategies that coordinate transmission timing, agents can adapt to challenges such as terrain-induced occlusions and electromagnetic interference, thereby reducing dependence on unreliable links [22], [23]. This framework is referred to as Task Allocation with Communication Coordination (TACC). To enable policy learning, we design a Multi-Agent Reinforcement Learning (MARL) environment based on the Partially Observable Markov Decision Process (POMDP), where the core components are defined as follows:

Observation. We define the local observation of agent j at local step t as $o_{j,t}$, where each observation component is defined as follows: Referring to the work by [23], we use the Bron-Kerbosch feature (BK) feature, value of message (VoM) feature, and normalized time step, and introduce the channel access feature, which are collectively used as the agent's observation of the current communication state.

Action. We define the action of agent j at time step t as $a_{j,t}$, which is defined as communication strategy based on the gating mechanism, allowing the agent to choose the timing for data transmission adaptively. At time step t, agent j inputs its local observation $o_{j,t}$ into the policy network to obtain the communication action $a_{j,t} \in \{0,1\}$, which determines whether a message should be sent. When $a_{j,t} = 0$, the agent does not transmit. Conversely, the agent chooses to transmit.

State Transition. We view one round of the communication and computation process in task allocation of agent j as a time step t_j. During this step, the allocation algorithm undergoes one iteration consisting of one computation and one communication stage. In the computation stage, the agent removes conflicting tasks based on consistency conflict resolution rules and inserts new tasks to update the task allocation results. In the same time slot T_i, some agents that have completed their computations form the set of the available agent, denoted as $u \subseteq \mathbf{U}$. The action set a_{u,T_i} of all available agents is retrieved and utilized to identify the data transmission at T_i. In the communication stage, the TDMA protocol automatically allocates time slots to all agents who need to communicate. The next global state $S_{T_{i+1}}$ are obtained through the state transition function $\mathscr{T}(S_{T_{i+1}}, a_{u,T_i})$.

Reward Function. We introduce a cooperative relationship to coordinate communication timing and define the reward function, which can be achieved by constructing a shared goal among the agents based on the communication strategy of the gating mechanism. To this end, we sum the changes in the task conflicts across all agents to form a global shared reward.

$$r_{T_i} = \sum_{j=1}^{n} (\triangle \mathbf{p}_{j,T_i} - \triangle \mathbf{p}_{j,T_{i+1}}) \tag{3}$$

Where $\triangle \mathbf{p}_{j,T_i} - \triangle \mathbf{p}_{j,T_{i+1}}$ represents the change in the average task conflicts for agent j during time slots $T_{i+1} - T_i$. $\triangle \mathbf{p}_{j,T_i}$ represents the average task conflicts between agent j and the others at the time slot T_i:

$$\triangle \mathbf{p}_{j,T_i} = \sum_{k=1}^{n} |\mathbf{p}_{j,T_i} \cap \mathbf{p}_{k,T_i}|/n \tag{4}$$

Where $\mathbf{p}_{j,T_i}$ represents the task list of agent j, and $\mathbf{p}_{k,T_i}$ represents the task list of another agent k. The more task conflicts in the environment, the larger the value of $\triangle\mathbf{p}_{j,T_i}$. To simplify notation, we define the reward r_{T_i} at global time slot T_i as r_t at the corresponding local time step t.

Transmission Constraint. In TACC, we extend the original POMDP by incorporating the minimization of communication overhead as an additional objective in policy optimization and introduce the transmission constraint to stabilize policy updates, reducing redundant communication in task allocation. The transmission constraint establishes a threshold for communication bandwidth in the gated strategy, which represents the minimum communication overhead required to achieve conflict-free task allocation. Firstly, the transmission constraint is described as the communication overhead incurred by an agent based on the maximum transmission probability p_{sup} within a task allocation round:

$$c_{sup} = p_{sup}\mathbb{E}_{j\in\mathbf{U},t}[\sum_{j=0}^{|\mathbf{U}|}\sum_{t=0}^{\infty}E_{j,t}] \tag{5}$$

Where the $\mathbb{E}_{j\in\mathbf{U},t}[\sum_{j=0}^{|\mathbf{U}|}\sum_{t=0}^{\infty}E_{j,t}]$ represents the average bandwidth required of data sent by each agent over a step, which can be estimated using statistical metrics. Next, the maximum transmission probability p_{sup} can be derived from the relevant parameters of TDMA and Shannon's theorem [24]:

$$p_{sup} = \text{clip}(\frac{8TZ}{10^{-3}\log(2\pi\lambda\sigma^2)N(N-1)Ld}, 0, 1) \tag{6}$$

where T represents the maximum number of time slots, d represents each time slot has an occupancy time, Z represents the maximum amount of data that can be transmitted per time slot, and L represents the number of symbols that Each message contains.

B. Asynchronous Experience Aggregation Method

In the CTDE, we need to collect training data from each agent, including interaction data between the agents and the environment, such as states, observations, actions, rewards, etc. Then, the interaction data is concatenated in chronological order into trajectories. All agents simultaneously make decisions and collect interaction data in the synchronous environment. As shown in Fig. 1(a), the integrated trajectory collected by agents #1, #2 and #3 are:

$$\begin{aligned}&[(s_0, o_{1,0}, o_{2,0}, o_{3,0}, a_{1,0}, a_{2,0}, a_{3,0}),\\ &\ldots, (s_n, o_{1,n}, o_{2,n}, o_{3,n}, a_{1,n}, a_{2,n}, a_{3,n})].\end{aligned} \tag{7}$$

However, most realistic real-time cooperative tasks require that agents act without waiting for other agents to avoid disastrous consequences. Existing methods adapt the original MARL designed for synchronous scenarios to implement training in asynchronous environments, and these methods can be categorized into three types. The padding method transforms asynchronous interaction data into synchronous ones via padding blank slots to apply existing MARL. As shown in Fig. 1(b), agent #1 and agent #3 are not making decisions in the time slot T_2, the placeholder actions $a'_{1,2}$ and $a'_{3,2}$ are padded. The discarding method makes agents collect data of other agents only when they make decisions but discard the data of others while executing their actions. As shown in Fig. 1(c), agent #1 ignores the actions of other agents, resulting in a blank action at the time slot T_2. The splicing method splices asynchronous interaction data from different agents into data at the same time step. As shown in Fig. 1(d), the interaction data of agent #1 and agent #2 are naturally spliced at time slot T_3. The trajectories of agent #1, #2, and #3 are respectively written as follows:

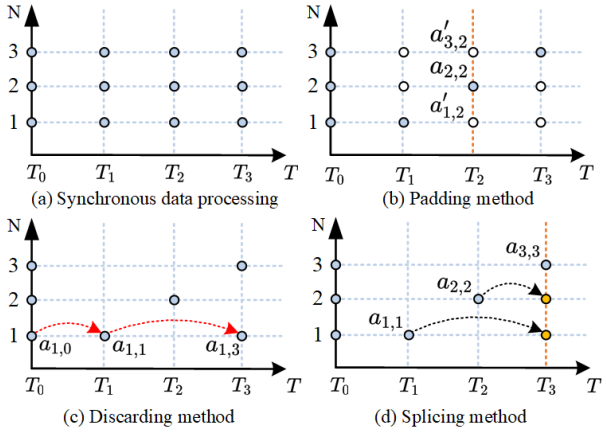

Fig. 1. illustration of different asynchronous MARL methods. Blue/yellow circles depict collected decision data. White ones depict padding actions. (a) Trajectory data collected by the Agent #1, #2, and #3 in a synchronous reinforcement learning environment. (b) Agent #1 and Agent #3 pad the decision data for the time slot T_2. (c) Agent #1 discards the decision at time slot T_2. (d) Agent #1 and Agent #2 concatenate the decisions from time slot T_1 and t_2 into time slot T_3.

$$\begin{aligned}&[(s_0, o_{1,0}, a_{1,0}), (s_1, o_{1,1}, a_{1,1}), \ldots]\\ &[(s_0, o_{2,0}, a_{2,0}), (s_2, o_{2,2}, a_{2,2}), \ldots]\\ &[(s_0, o_{3,0}, a_{3,0}), (s_3, o_{3,3}, a_{3,3}), \ldots]\end{aligned} \tag{8}$$

In comparison, the splicing method directly combines interaction data collected from different agents with minimal processing. Therefore, we adopt the splicing method for use in the TACC framework.

C. Multi Objective Constrained Policy Optimization

Within the POMDP framework, TACC's goal is cast as a multi-objective MARL task, which can be equivalently viewed as a constrained MARL problem with multiple constraints. Two leading solution families exist: the primal-dual method and constrained policy optimization [25]. In TACC, we apply constrained policy optimization directly to the policy gradient via a Lagrangian loss, stabilizing gated communication early in training and boosting sample efficiency and convergence. To operationalize this, we merge constrained policy optimization with proximal policy optimization, yielding our novel MARL algorithm, MOCPO. The detailed MOCPO implementation follows:

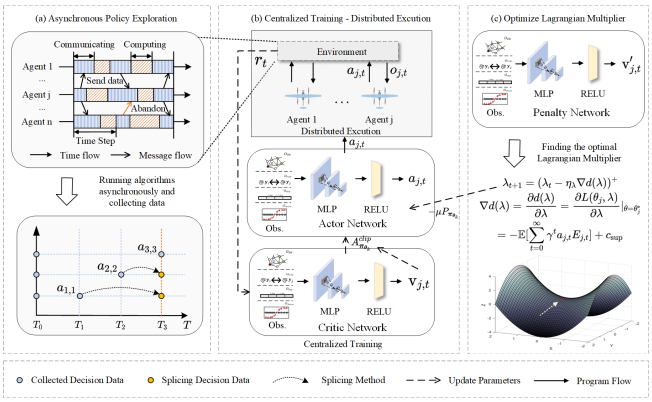

Fig. 2. The architecture of MOCPO: (a) Asynchronous data collection by splicing agents' trajectories; (b) CTDE framework with processed data for centralized training, where a Lagrange multiplier regulates actor loss; (c) Penalty network optimizes the Lagrange multiplier from distributed execution experiences.

Algorithm 1 Multi-Objective Constrained Policy Optimization Algorithm

Initialize: Central critic network θ_{critic}, Central actor network θ_{actor}, penalty network $\theta_{penalty}$ and Lagrangian multiplier λ.

for each training episode *eps* **do**
  $S = S_0$, $t = 0$ for each agent i
  for $t = 1 \leftarrow T_j$ **do**
    for all available agent j **do**
      $v_{j,t} = V^{\pi}_{\theta_{critic}}(o_{j,t})$
      $v'_{j,t} = V^{\pi}_{\theta_{penalty}}(o_{j,t})$
      Sample $a_{j,t}$ from $\pi(o_{j,t})$
    end for
    Execute the joint action $(a_{1,t}, a_{2,t}, ..., a_{n,t})$
    Get reward $r_{j,t}$ and next state S_{t+1}
    $S = S_{t+1}$
  end for
  Add episode to replay buffer $(v_{j,t}, v'_{j,t}, o_{j,t}, a_{j,t}, r_{j,t})$
  Splicing interaction data from different time slots using splicing methods
  Calculate the constraint value of communication bandwidth c_{sup} with (18)
  Compute the Lagrangian loss $L(\theta,\lambda) = A^{clip}_{\pi_{\theta_k}}(o_{j,t}, a_{j,t}) - \lambda P_{\pi_{\theta_k}}(o_{j,t}, a_{j,t})$
  for training batch d in *batchs* **do**
    for t = 1 to T **do**
      Update central critic network θ_{critic} with (26)
      Update central actor network θ_{actor} with (28)
      Update penalty network $\theta_{penalty}$ with (29)
      Compute the Lagrangian multiplier $\lambda_{t+1} = (\lambda_t - \eta_\lambda(-v'_{j,t} + c_{sup}))^+$
    end for
  end for
end for

1) Optimization Algorithm: The idea of MOCPO is to introduce transmission penalties during the optimization of communication strategies to stabilize the training gradient, balancing task conflicts and communication overhead by iteratively updating the Lagrangian multipliers. On this basis, MOCPO maximizes the reward while simultaneously limiting the constraints of transmission cost, updating the parameters within the constrained policy domain. To this end, the optimization problem of communication strategy in the algorithm can be written as:

$$\begin{aligned} &\max_{\theta_k} J(\theta_k) = \mathbb{E}_{s\sim\rho_{\pi_{\theta_k}}}[A_{\pi_{\theta_k}}(o_{j,t}, a_{j,t})] \\ &\theta \\ &\text{s.t. } C(\theta_k) = \mathbb{E}_{s\sim\rho_{\pi_{\theta_k}}}[\sum_{t=0}^{\infty}\gamma^t a_{j,t}E_{j,t}] \le c_{\text{sup}}\,. \end{aligned} \tag{9}$$

Where $A_{\pi_{\theta_k}}$ represents the advantage value of the observation action pair $(o_{j,t}, a_{j,t})$ of agent j under the policy function π. Due to the difficulty in directly solving this problem, we approximate the function by first-order expansion of the optimization objective and cost constraints using Taylor's formula. Using first-order Taylor expansion, we can obtain formal representations of the objective function and constraint function:

$$\begin{aligned} &\mathbb{E}_{s\sim\rho_{\pi_{\theta_k}}}[A_{\pi_{\theta_k}}] \approx \mathbb{E}_{s\sim\rho_{\pi_{\theta_k}}}[A_{\pi_{\theta_k}}] + \nabla_\theta J(\theta_k)^\top \Delta\theta \\ &\mathbb{E}_{s\sim\rho_{\pi_{\theta_k}}}[\sum_{t=0}^{\infty}\gamma^t a_{j,t}E_{j,t}] \approx \mathbb{E}_{s\sim\rho_{\pi_{\theta_k}}}[\sum_{t=0}^{\infty}\gamma^t a_{j,t}E_{j,t}] + \nabla C(\theta_k)^\top \Delta\theta \end{aligned} \tag{10}$$

In the formula, the constant term is omitted as it does not affect the parameter solution. We define $\mathbf{b} := \nabla_\theta J(\theta_k)$ as the gradient vector of the advantage function, $\mathbf{g} := \nabla_\theta C(\theta_k)$ as the gradient vector of the constraint function, and ε as the current remaining "constraint margin", where $\varepsilon := c_{\text{sup}} - C(\theta_k)$. The optimization problem can be written as:

$$\begin{aligned} &\max_{\Delta\theta} \quad \mathbf{b}^\top\Delta\theta \\ &\text{s.t.} \quad\quad \mathbf{g}^\top\Delta\theta \le \varepsilon \end{aligned} \tag{11}$$

By introducing the multipliers λ, we construct the Lagrangian function for this problem:

$$L(\Delta\theta, \lambda) = \mathbf{b}^\top\Delta\theta + \lambda(\mathbf{g}^\top\Delta\theta - \varepsilon). \tag{12}$$

At this point, the problem is transformed into an unconstrained optimization problem, and the goal of the agent becomes to maximize the Lagrangian function value $L(\Delta\theta, \lambda)$. Due to the difficulty in calculating $\Delta\theta^*$ through the optimal λ value, we approximate $\Delta\theta^*$ using gradient descent:

$$\Delta\theta^* = -\eta(\mathbf{b} + \lambda\mathbf{g}) \tag{13}$$

Where η is the step size for policy gradient update, and $\Delta\theta^*$ can be adaptively solved using the Adam optimizer [26]. For the policy network θ_{actor}, the parameters are optimized by minimizing the Lagrangian loss $L(\theta, \lambda)$,

$$\theta_{\text{actor}} \leftarrow \arg\max_{\theta_{\text{actor}}} \mathbb{E}_{o_{j,t}, a_{j,t}\sim\pi_{\theta_k}}[A^{clip}_{\pi_{\theta_k}}(o_{j,t}, a_{j,t}) - \lambda P_{\pi_{\theta_k}}(o_{j,t}, a_{j,t})] \tag{14}$$

Where $A^{clip}_{\pi_{\theta_k}}$ represents the advantage value of the current action state pair, and $P_{\pi_{\theta_k}}$ represents the penalty value for the communication overhead of the agent under the current policy. To avoid excessive differences in policy distribution before and after parameter updates, we introduce the KL divergence as a weight factor ω for the advantage value, and set an upper bound for the advantage value through a truncation method. Together, these two factors limit the range of policy gradient values to ensure stability during policy updates.

$$A^{clip}_{\pi_{\theta_k}}(o_{j,t},a_{j,t}) = \min(\omega \times A_{\pi_{\theta_k}}(o_{j,t},a_{j,t}), \\ \text{clip}(\omega, 1-\varepsilon, 1+\varepsilon) \times A_{\pi_{\theta_k}}(o_{j,t},a_{j,t})) \tag{15}$$

Where $\omega = \bar{D}_{\text{KL}}(\pi_{\theta_{k+1}}, \pi_{\theta_k})$ and $\pi_{\theta_{k+1}}$ and π_{θ_k} represent the policy distributions of the agent before and after updating, respectively. For the evaluation network, we calculate the training gradient under the condition of minimizing the mean squared distance between the state value function V^{π} and the global consistency reward r to update the evaluation network parameters:

$$\theta_{\text{critic}} \leftarrow \arg\min_{\theta_{\text{critic}}} \mathbb{E}_{o_{j,t},a_{j,t}\sim\pi_{\theta_k}}[(V^{\pi}(o_{j,t}) - r)^2] \tag{16}$$

During the training process, $P_{\pi_{\theta_k}}$ obtains the penalty value used to evaluate the communication overhead through sampling, which is difficult to achieve gradient conduction in neural network training. Therefore, we construct a penalty network $\theta_{penalty}$ to approximate $P_{\pi_{\theta_k}}$, which is used to estimate the required bandwidth value $V_{\theta_{\text{penalty}}}(o_{j,t}) \approx \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t a_{j,t} E_{j,t}]$ for communication under different observations of the agent. The parameters of this network are also calculated using the mean squared error distance:

$$\theta_{\text{penalty}} \leftarrow \arg\min_{\theta_{\text{penalty}}} \mathbb{E}_{o_{j,t},a_{j,t}\sim\pi}[(V^{\pi}_{\theta^{\pi}_{\text{penalty}}}(o_{j,t}) - a_{j,t}E_{j,t})^2] \tag{17}$$

To achieve simultaneous optimization of multiple objectives, we aim to maximize the advantage value while also increasing the influence of λ on the policy network updates as much as possible. Therefore, during the training of policy parameters, we obtain the gradient by differentiating λ with respect to the Lagrangian function $L(\theta,\lambda)$ to iteratively find the optimal λ^*:

$$\begin{aligned} \lambda_{t+1} &= (\lambda_t - \eta_\lambda \nabla d(\lambda))^+ \\ \nabla d(\lambda) &= \frac{\partial d(\lambda)}{\partial \lambda} = \frac{\partial L(\theta_j,\lambda)}{\partial \lambda}|_{\theta=\theta^*_j} \\ &= -\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t a_{j,t} E_{j,t}] + c_{\text{sup}} \end{aligned} \tag{18}$$

In practical implementations, $-\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t a_{j,t} E_{j,t}]$ is substituted with the communication overhead $P_{\pi_{\theta_k}}(o_{j,t},a_{j,t})$ of the agent. Based on this method, we iteratively update the policy network, evaluation network, penalty network, and Lagrangian multiplier λ through exploration in the policy domain.

TABLE I
PARAMETERS OF SIMULATION

Parameter	Value
Frame Length	22 ms
Number of Slots	64
Slot Length	345 λs
Maximum Transmission Unit (MTU)	1000 bits

IV. SIM-TO-REAL EXPERIMENT

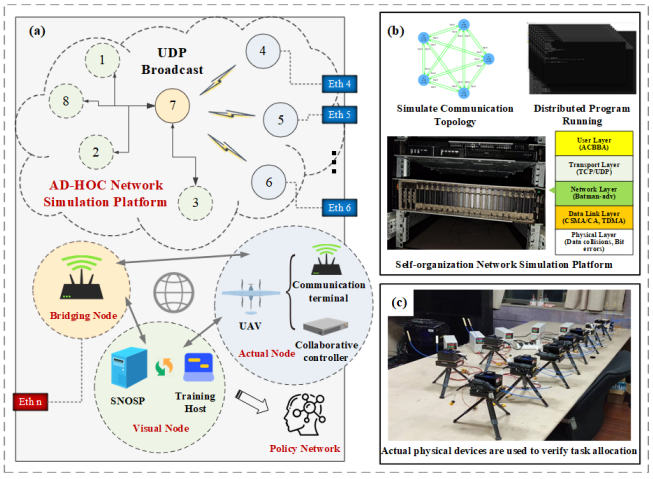

Fig. 3. The HIL training and validation environment. (a) Ad-hoc network is composed of virtual node, bridge node and actual node. Green represents virtual nodes, orange represents bridge nodes, and blue represents actual nodes. (b) SNOSP can simulate the communication topology and distributed program operation of UAV swarm. (c) Validating the policy in the task allocation of the actual UAV swarm.

A. Environment and Experimental Settings

Validating the performance of TACC, this paper builds a hardware-in-the-loop (HIL) training and validation environment based on the work of [27], which is used to verify the distributed task allocation of the UAV swarm. The HIL environment is shown in Fig. 3, which is mainly composed of a Self-organization Network Simulation Platform (abbreviated as SNOSP), Actual Physical Devices (abbreviated as APD), Training Host (abbreviated as TH), etc. The SNOSP mainly includes three types of communication nodes: virtual nodes, bridge nodes, and actual nodes (Fig. 3(a)). The virtual node is a simulated networking device built inside SNOSP and runs the 5-layer protocols APD uses (Fig. 3(b)). The bridge node is a node that connects SNOSP and APD, whose physical layer protocol runs on external APD, while the other 4-layer protocols run on SNOSP. The actual node is APD outside SNOSP, where the information of external TH is transmitted to the bridge node wirelessly. The HIL network simulation system composed of N real nodes and X virtual nodes can truly simulate the swarm networking communication, and as such, the $N+X$ method has good scalability of communication nodes (Fig. 3(c)).

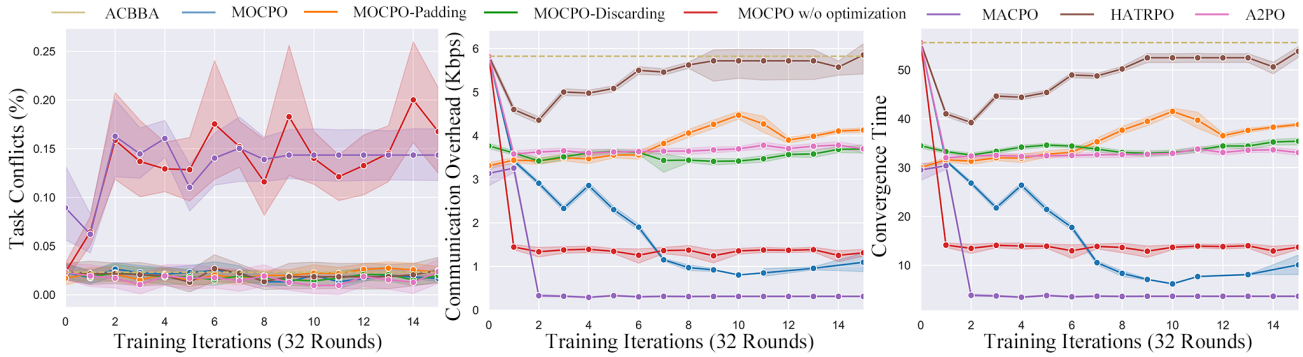

Fig. 4. Comparison of task conflicts, communication overhead, and convergence time under different MARL methods in 30-UAV scenarios.

B. Evaluation Metrics for the Distributed Method

The distributed method are primarily evaluated based on task conflicts, communication overhead, and convergence time.

(1) Task conflicts - ratio of the number of tasks assigned to more than one agent to the total number of tasks. Then task conflicts $R_{conflicts}$ is defined as:

$$R_{conflicts} = \frac{|\{k \mid \sum_j x_{jk} > 1\}|}{m}. \tag{19}$$

Where $\{k \mid \sum_j x_{jk} > 1\}$ is the set of all tasks with more than one allocation. m is the number of tasks required to allocate.

(2) Communication overhead - during the communication stage of distributed task allocation, the average bandwidth required for agent communication. The communication overhead $B_{overhead}$ is follow as:

$$B_{overhead} = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} B_t. \tag{20}$$

Where B_t is the total bytes exchanged by all agents in step t, t_{max} represents the maximum number of steps.

(3) Convergence time - time taken for task allocation to converge to a conflict-free allocation. This metric is defined in terms of rounds of task allocation methods or rounds of communication. The convergence time is the smallest step t^* such that

$$t^* = \min\{t \mid \mathscr{C}(t) = \text{ true } \}. \tag{21}$$

Where $\mathscr{C}(t)$ is a boolean indicator that the allocation is conflict-free after step t.

C. Training Performance

To verify the effectiveness of MOCPO, we collected data and trained policies in task allocation scenarios with 30 networked UAVs. MACPO, HATRPO, and A2PO were used as baseline methods, with MACPO and HATRPO being state-of-the-art constrained multi-agent reinforcement learning methods, while A2PO is an improved version of MAPPO. Task conflicts, communication overhead, and convergence time were used as evaluation metrics.

As shown in Fig. 4, the results confirm the best training efficiency of the proposed method's overall state-of-the-art approaches.Compared to ACBBA, MOCPO reduces the task conflict rate by 28.13%, and decreases the communication overhead and convergence time by 81.44% and 84.61% respectively, achieving an optimal trade-off among multiple optimization objectives. However, the performance of MACPO, HATRPO, and A2PO is unsatisfactory. The task conflict of MACPO is excessively high (+520.77% vs. ACBBA), and HATRPO incurs significant communication overhead without reducing the conflict rate (+17.69% vs. ACBBA). Meanwhile, A2PO has trained a suboptimal solution that simultaneously reduces communication overhead and convergence time to a certain extent (-36.59% and -40.49% vs. ACBBA) while maintaining the task conflict rate unchanged (+2.59% vs. ACBBA). The results demonstrate the benefits of constrained policy optimization, which ensures monotonic improvement and constraint satisfaction of the communication strategy, leading to superior training performance.

D. The Impact of The Asynchronous Experience Collection and Concatenation

To assess the impact of asynchronous experience collection and concatenation, we trained communication strategies using MOCPO-Padding and MOCPO-Discarding, which only modified trajectory processing on top of MOCPO. As shown in Fig.4, all three methods maintained low task conflicts (1.66%, 2.27%, 1.95% vs. 2.31% for ACBBA), but Padding and Discarding produced suboptimal solutions. Compared with ACBBA, they reduced communication overhead by 29.03% and 36.59% and convergence time by 30.14% and 36.30%. In contrast, the concatenation method significantly lowered both communication overhead and convergence time, likely because it accounts for other agents' strategies within each time step, enabling more accurate credit assignment and improving sample efficiency in MARL.

E. Ablation Study

The constrained policy optimization method enables MOCPO to achieve multiple objectives during the training strategy, reducing task conflicts and communication

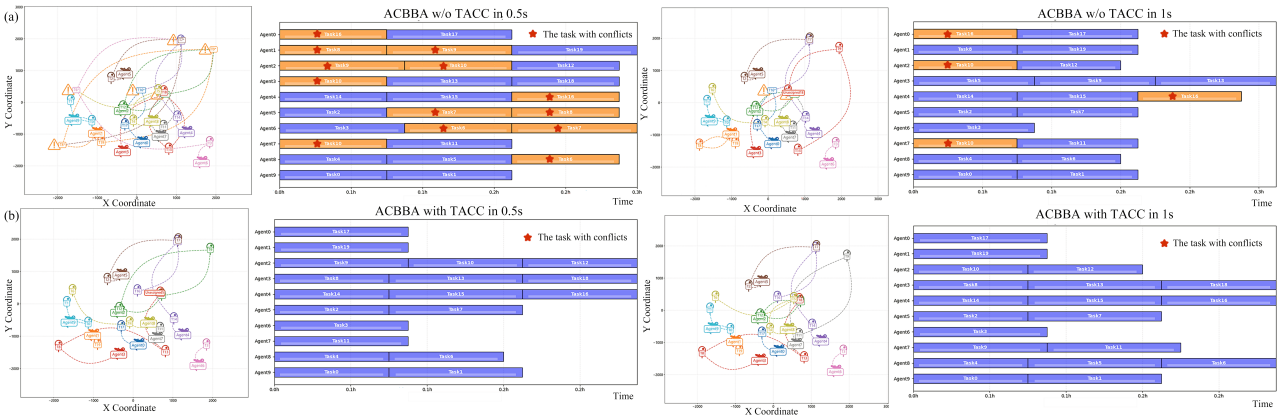

Fig. 5. Search and rescue experiments with 10 UAVs, 20 survivors are located randomly, and are served in the order of their location in the assigned path. Time scheduling for tasks is shown when ACBBA w/o communication policy (a) or ACBBA with communication policy (b) is applied for task scheduling.

overhead. To verify the impact of the constrained policy optimization, we conducted ablation in the environment with 30 agents. As shown in Fig. 4, the MOCPO without optimization struggles to learn an effective communication strategy, exhibiting only single-objective training results. While the ablated MOCPO significantly reduces communication overhead and convergence time by up to 89.51% and 88.03% compared with ACBBA, task conflicts increase by 625.54%. Excessive task conflicts and high communication overhead make it difficult for a UAV swarm to complete task allocation, highlighting the importance of multi-objective optimization. The results demonstrate that the ablated MOCPO is misdirected in its gradient updates due to single reward feedback under conditions where communication constraints are absent, making it challenging to train effective strategies in a vast policy space.

V. FLIGHT EXPERIMENT

Fig. 6. Fei Long-1700 UAV swarm executes distributed task allocation algorithm in the search and rescue task.

A. The Configuration of the UAV swarm

To verify applications of TACC in the real-world UAV swarm, we simulated a search and rescue mission scheduling scenario and conducted a flight test with 10 UAVs. As shown in Fig. 6, the UAV swarm consists of individual UAVs, each being a hybrid-wing UAV (Fei Long-1700) equipped with a collaborative controller and a communication terminal. The collaborative controller is constructed based on the Rockchip RK3588 SOC, which can carry task allocation algorithms and small-scale neural networks through the Linux kernel. The communication terminal is responsible for inter-UAV communication within the swarm and air-to-ground data transmission, enabling swarm information sharing and telemetry data feedback. Under the construction of a sim-to-real ad-hoc network, the ground station can connect with ten UAVs through the communication terminal, enabling real-time monitoring of aircraft parameter changes and flight status.

B. The Search and Rescue Scenario

Distributed task allocation is often used for time-sensitive tasks, where a shorter completion time typically yields a higher reward. For instance, in a search and rescue scenario, a group of agents should move to the assigned targets and provide medical supplies, food, or transportation as soon as possible [23], [13], [27]. In this task scenario, the task reward can be represented by a time-discounted reward, where the total reward of tasks assigned to agent j is:

$$S_j^{\mathbf{p}_j} = \sum_{k=1}^{|\mathbf{K}|} \varphi^{\tau_j^k(\mathbf{p}_j)} \bar{c}_k \tag{22}$$

Where $\varphi \in (0,1]$ is a discount factor and $\tau_j^k(\mathbf{p}_j)$ is the time when agent j arrives at task k along the allocated path $\mathbf{p}_j$. $\bar{c}_k$ is the static reward for task k. The exponential function computes the task reward:

$$\bar{c}_k = e^{-d_k} \tag{23}$$

Where the d_k is the distance traveled by agent j to arrive at target k. Furthermore, We can compute the time $\tau_j^k(\mathbf{p}_j)$:

$$\tau_j^k(\mathbf{p}_j) = d_k v_j^{-1} + (\zeta_k - 1)\delta \tag{24}$$

Where v_j is the speed of agent j, $\zeta_k \in \{1,2,...,N_j\}$ is the order of targets and δ is the time required for serving each target.

C. Time Scheduling of The Flight Experiment

We designed the flight experiment in a search and rescue scenario to validate TACC's efficiency under real-world task allocation. In the experiment, the operational area is defined as a square region no larger than 4 km on each side, and the task reward is computed based on the UAV's initial position, speed, fuel level, and the task's execution time, start time, and deadline. In this setup, the UAV swarm must service multiple tasks with strict deadlines to minimize missed tasks. Furthermore, we set the collaborative controller to complete communication and task scheduling within 0.5s and 1s.

The result shows that TACC completed the conflict-free allocation of all tasks within 1 second. In contrast, ACBBA only achieved conflict-free assignments for 17 tasks, illustrating that TACC's timelessness is superior to ACBBA's. From the task path in Fig. 5(a), ACBBA generated thirteen path conflict points within 0.5 seconds, four path conflict points, and one unassigned point within 1 second. From the task path in Fig. 5(b), TACC generated only one unassigned point within 0.5 seconds and resolved all conflicts within 1 second. These results indicate that TACC, compared to ACBBA, transmitted more valuable messages through communication policy coordinating within the same time slots, resulting in superior scheduling outcomes when they optimize the consistent time-sensitive reward function.

VI. CONCLUSIONS

This paper optimizes communication policies for asynchronous distributed task allocation by formulating the problem as a POMDP with a transmission constraint to guide exploration. We propose MOCPO, a MARL method using Lagrangian loss with transmission penalties to reduce communication overhead and convergence time. Experiments in a Hardware-in-the-Loop environment and sim-to-real setups show that TACC outperforms state-of-the-art baselines (MACPO, HATRPO, A2PO) and achieves superior performance in search-and-rescue trials. Flight experiment further confirms the effectiveness and timelessness of the proposed TACC within the ten-UAV swarm in the search and rescue.

REFERENCES

[1] E. Debie, K. Kasmarik, and M. Garratt, "Swarm robotics: A survey from a multi-tasking perspective," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–38, 2023.

[2] M. Schranz, G. A. Di Caro, T. Schmickl, W. Elmenreich, F. Arvin, A. Şekercioğlu, and M. Sende, "Swarm intelligence and cyber-physical systems: concepts, challenges and future trends," *Swarm and Evolutionary Computation*, vol. 60, p. 100762, 2021.

[3] E. Soria, F. Schiano, and D. Floreano, "Predictive control of aerial swarms in cluttered environments," *Nature Machine Intelligence*, vol. 3, no. 6, pp. 545–554, 2021.

[4] M. Dorigo, G. Theraulaz, and V. Trianni, "Reflections on the future of swarm robotics," *Science robotics*, vol. 5, no. 49, p. eabe4385, 2020.

[5] S. Poudel and S. Moh, "Task assignment algorithms for unmanned aerial vehicle networks: A comprehensive survey," *Vehicular Communications*, vol. 35, p. 100469, 2022.

[6] J. Tang, G. Liu, and Q. Pan, "A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1627–1643, 2021.

[7] J. C. Amorim, V. Alves, and E. P. de Freitas, "Assessing a swarm-gap based solution for the task allocation problem in dynamic scenarios," *Expert Systems with Applications*, vol. 152, p. 113437, 2020.

[8] J. Tang, X. Chen, X. Zhu, and F. Zhu, "Dynamic reallocation model of multiple unmanned aerial vehicle tasks in emergent adjustment scenarios," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 2, pp. 1139–1155, 2022.

[9] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and distributed control of a large-scale swarm of autonomous agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[10] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, S.-Y. Liu, J. P. How, and J. Vian, "Decentralized control of multi-robot partially observable markov decision processes using belief space macro-actions," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 231–258, 2017.

[11] C. O. Aguilar and B. Gharesifard, "Almost equitable partitions and new necessary conditions for network controllability," *Automatica*, vol. 80, pp. 25–31, 2017.

[12] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu, and J. W. Herrmann, "Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3770–3776, IEEE, 2020.

[13] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[14] L. Johnson, H.-L. Choi, and J. P. How, "The hybrid information and plan consensus algorithm with imperfect situational awareness," in *Distributed Autonomous Robotic Systems: The 12th International Symposium*, pp. 221–233, Springer, 2016.

[15] C. H. Papadimitriou, "Computational complexity," in *Encyclopedia of computer science*, pp. 260–265, 2003.

[16] R. Wang, X. He, R. Yu, W. Qiu, B. An, and Z. Rabinovich, "Learning efficient multi-agent communication: An information bottleneck approach," in *International Conference on Machine Learning*, pp. 9908–9918, PMLR, 2020.

[17] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in uav-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[18] M. L. Betalo, S. Leng, H. N. Abishu, F. A. Dharejo, A. M. Seid, A. Erbad, R. A. Naqvi, L. Zhou, and M. Guizani, "Multi-agent deep reinforcement learning-based task scheduling and resource sharing for o-ran-empowered multi-uav-assisted wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 9247–9261, 2023.

[19] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous decentralized task allocation for dynamic environments," in *Infotech@ Aerospace 2011*, p. 1441, 2011.

[20] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[21] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi, "Learning to schedule communication in multi-agent reinforcement learning," *arXiv preprint arXiv:1902.01554*, 2019.

[22] G. Hu, Y. Zhu, D. Zhao, M. Zhao, and J. Hao, "Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 3966–3978, 2021.

[23] S. Raja, G. Habibi, and J. P. How, "Communication-aware consensus-based decentralized task allocation in communication constrained environments," *IEEE Access*, vol. 10, pp. 19753–19767, 2021.

[24] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[25] C. Tian, A. Liu, G. Huang, and W. Luo, "Successive convex approximation based off-policy optimization for constrained reinforcement learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1609–1624, 2022.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] R. Chen, J. Li, Y. Chen, and Y. Huang, "A distributed scheduling method for networked uav swarm based on computing for communication," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11071–11078, IEEE, 2023.