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

Precedence-Aware Multi-UAV Task Allocation with an Attention-Based Reinforcement Learning Framework

Xurui Liu[1,2]

***Abstract*— Multi-UAV coordination is critical for complex real-world applications, but these missions are often constrained by intricate causal dependencies between tasks, alongside strict UAV energy and return-to-base constraints. Existing methods, ranging from exact solvers to standard deep reinforcement learning approaches, struggle to scale with the combinatorial complexity of this problem and often fail to effectively represent the underlying logical task structures. To address this gap, we propose the Causal-Channel Transformer for Joint Allocation (C2T-JA), an end-to-end reinforcement learning framework. The core of C2T-JA is a dual-branch hybrid attention encoder that explicitly constructs and reasons over multi-hop, disentangled causal channels, effectively decoupling logical dependencies from spatial task features. Building on this rich representation, a context-aware decoder generates a globally coordinated joint action for the entire team. We evaluated C2T-JA against established baselines, including an exact solver (Gurobi), a conventional heuristic (OR-Tools), and a leading learning-based approach (AM-joint), on procedurally generated benchmarks of varying scales and dependency structures. The results demonstrate that our approach consistently produces higher-quality solutions, measured by task completion rates, while reducing decision times by several orders of magnitude, particularly in large-scale scenarios.**

I. INTRODUCTION

Due to the inherent constraints on the endurance and operational range of a single Unmanned Aerial Vehicle (UAV), coordinating multi-UAV teams for complex tasks has become an active research area in modern robotics, with applications in search-and-rescue, agricultural monitoring, and environmental sensing [1], [2], [3]. In practice, these tasks are rarely independent. As shown in Fig. 1, they are governed by strict temporal constraints and dependencies, which dictate that a task cannot begin until its prerequisites are completed. This necessitates that each UAV make decisions based on the team's global state to pursue team-level optimality. Furthermore, the limited endurance of UAVs adds another constraint, requiring the team to maximize task completion within a finite energy budget, while ensuring every UAV can return to base.

This combination of causal dependencies with energy and return-to-base constraints gives rise to a highly complex problem, which falls squarely into the category of Multi-Robot Task Allocation with Temporal and Ordering Constraints (MRTA/TOC) [4]. Existing methods show notable limitations in handling such combined constraints. Exact methods like Mixed-Integer Programming (MIP) [5], [6] can find optimal solutions for small-scale problems, but their exponential computational complexity makes them impractical beyond modest instance sizes. While computationally faster, heuristic and meta-heuristic algorithms [7] rely on handcrafted rules, which limits their ability to generalize across diverse and complex dependency structures. Similarly, distributed auction algorithms [8], [9] support decentralized decision-making but are susceptible to getting stuck in local optima when faced with multi-level dependencies that require global coordination. Recently, learning-driven approaches, particularly Deep Reinforcement Learning (DRL), have emerged as a promising direction. Among these, attention-based architectures have demonstrated notable performance [10], [11]. While these methods show potential for efficient and generalizable decision-making, their standard attention structures can create a representational bottleneck when handling intricate dependencies, and struggle to decouple tasks' physical attributes from their logical dependencies. This reveals the need for a decision-making algorithm that explicitly models and leverages intricate causal dependencies.

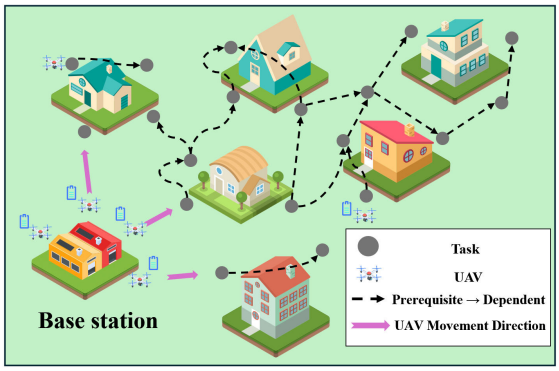


Fig. 1. An illustration of the multi-UAV task allocation problem with causal dependencies. A team of UAVs departs from the base station to perform spatially distributed tasks, which are interconnected by prerequisite constraints (dashed arrows) dictating the execution order.

To address this problem, we propose the Causal-Channel Transformer for Joint Allocation (C2T-JA), an end-to-end DRL framework with a hybrid attention mechanism. Its core is a dual-branch attention encoder that uses a causal extension mechanism to explicitly construct multi-order, disentangled dependency channels. By processing feature correlations and logical dependencies in two separate branches, this design effectively decouples these information sources, generating richer representations that encode the complex task logic. Building on these, a context-aware decoder uses

[1] School of Robotics and Automation, Nanjing University, Suzhou, 215163, China (e-mail: xrl250444@gmail.com).
[2] School of Aeronautics and Astronautics, University of Electronic Science and Technology of China, Chengdu, 611731, China.

a history attention module that leverages representations of future candidate tasks to attend to relevant historical information, constructing dynamic UAV representations. Combined with cross-attention and a dynamic multi-constraint mask, the decoder directly generates a globally feasible joint action for the entire team. We evaluated C2T-JA against established approaches, including the OR-Tools VRP solver [12], the Gurobi MIP solver [13], and AM-joint (adapted from AM [10]), on randomly generated instances of varying scales (e.g., 80 to 200 tasks) and varying dependency structures. The results show that C2T-JA yields solutions of higher quality, as evidenced by greater task completion rates and shorter decision times in most scenarios.

II. RELATED WORK

The MRTA/TOC problem involves the assignment and scheduling of tasks for multiple robots under time windows, precedence, and synchronization constraints [4]. The core challenge lies in coordinating complex temporal dependencies and scheduling. It is a generalization of the classic Vehicle Routing Problem (VRP) for multi-agent scenarios, further complicated by real-world operational factors like strict energy budgets.

Building on the above problem setting, prior work falls into three strands. First, while exact methods (e.g., MIP) offer theoretical guarantees of optimality, they scale poorly as instance size or constraint density grows. For instance, Bredström et al. [14] employed a branch-and-price framework and noted that the pricing subproblem in their column-generation approach becomes especially challenging under complex dependency structures. Similarly, Korsah et al. [15] showed that their set-partitioning formulation increases solver difficulty, as handling intricate precedence constraints leads to a densification of the constraint matrix. Second, heuristic algorithms trade theoretical optimality for computational speed. While efficient, heuristic algorithms do not guarantee global optimality, and their performance is often constrained by predefined search strategies that are sensitive to initialization. For example, Jones et al. [16] developed a local search strategy on time-extended graphs and highlighted that its performance is confined by the definition of the search neighborhood. Similarly, Bischoff et al. [7] proposed a two-stage method that combines greedy construction with local improvements, but they reported that under strong collaborative constraints, its improvement operators struggle to escape local optima when navigating the fragmented feasible region. Finally, distributed auction algorithms [17], [18] enhance scalability through decentralized decision-making. In practice, these algorithms rely on a greedy bidding mechanism that uses local price information. This approach struggles when strong collaborative constraints give rise to complex, non-submodular team utility. Under such conditions, the mechanism can misestimate the true combinatorial value of long precedence chains, ultimately yielding globally suboptimal assignments.

While DRL offers a promising direction for MRTA/TOC, current approaches still grapple with two fundamental challenges: representing causal dependencies and maintaining global consistency. The first of these manifests as a critical representational bottleneck for causal dependencies. Although Graph Neural Networks (GNNs) are a natural choice for modeling task relationships, classic models such as GCN [19] and GAT [20] are often hampered by over-smoothing and training instability when stacked to capture long-range causal dependencies. More importantly, higher-order GNNs [21], [22], [23], [25] designed to address this problem have inductive biases that primarily favor aggregating multi-hop neighborhood features or diversifying feature propagation. They lack mechanisms to explicitly encode the hierarchical (k-hop precedence) relationships inherent in a directed acyclic graph (DAG). Consequently, as standard attention mechanisms lack an explicit inductive bias for such hierarchical structures, attention-based DRL frameworks like AM [10] and CapAM [24] tend to produce myopic decisions when faced with dense, long-range causal dependencies. This highlights the need for an architecture that can inject the hierarchical causal structure of tasks as a strong inductive bias directly into the representation learning process.

The second major difficulty lies in coordinating joint decisions under heterogeneous constraints. Current DRL policies often fail to guarantee global consistency. This manifests in two ways. First, sequential decoding frameworks, such as the one used by AM, decide for one agent at a time. Faced with tightly coupled constraints, a single greedy decision can drastically shrink the subsequent feasible solution space, trapping the policy in a local optimum. Second, while traditional multi-agent reinforcement learning methods like Actor-Attention-Critic [26] excel at coordinating agent policies, their optimization objective is not the joint optimization of task-to-agent assignments, making it hard to ensure global feasibility under strict temporal constraints. Moreover, a systematic approach for managing multiple, heterogeneous constraints is often missing in existing work. Thus, a key research direction is the design of a DRL architecture capable of deeply representing complex causal structures while efficiently coordinating diverse and coupled constraints.

III. PROBLEM FORMULATION

A team of n autonomous UAVs, denoted by $U = \{u_1, u_2, \ldots, u_n\}$, departs from a single base station $\mathcal{T}_0$ to perform m spatially distributed tasks. Without loss of generality, the spatial coordinates of all tasks and the base station are assumed to lie within the unit square $[0, 1]^2$. We represent the base station and tasks as a unified set of nodes $\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m\}$, where each node $\mathcal{T}_i$ is associated with spatial coordinates (x_i, y_i) and a reward r_i representing its value ($\mathcal{T}_0$ has a reward of $r_0 = 0$). We define precedence constraints among tasks and model these relationships using a dependency matrix $D \in \{0, 1\}^{(m+1)\times(m+1)}$, where $D_{ij} = 1$ indicates that $\mathcal{T}_i$ is a prerequisite for $\mathcal{T}_j$. These dependencies collectively form a DAG. To maintain dimensional consistency, the matrix includes $\mathcal{T}_0$, but the base has no dependencies, i.e., $\forall i, j, D_{0j} = D_{i0} = 0$.

Each UAV $u_a \in U$ has a limited maximum flight distance $d_{\max,a}$. This constrains the UAV's operational range and mandates that any route ensures a return to $\mathcal{T}_0$. Additionally, each task can be performed at most once. Therefore, the optimization objective is to develop a cooperative policy that maximizes the team's total number of completed tasks, subject to precedence constraints, UAV energy budgets, and return-to-base requirements.

IV. Methodology

We formulate this problem as a Markov Decision Process (MDP) and introduce C2T-JA, our proposed end-to-end deep reinforcement learning framework. At its core, C2T-JA is a Transformer-based encoder-decoder network designed to learn a direct mapping from state to a joint team-wide action. The encoder is a dual-branch hybrid attention module that decouples the spatial attributes of tasks from their causal dependencies and learns their respective representations. The decoder is a context-aware module that generates a joint action for the entire team, satisfying all constraints. The following sections present the MDP formulation, network architecture, and training method.

A. RL formulation

a) State: At each decision step t, the state s_t provides a complete snapshot of the dynamic information needed for decision-making. The state is defined by the tuple $s_t = (Z_t, \mathcal{H}_t, M_t)$. The first component, $Z_t = \{z_{u,t}\}_{u=1}^{n}$, represents the UAV team's status, where each element $z_{u,t} = (p_{u,t}, d_{u,t})$ details UAV u's current location $p_{u,t} \in \mathbb{R}^2$ and remaining flight distance $d_{u,t} \in \mathbb{R}^+$. Initially, this distance is set to the UAV's maximum range, $d_{u,0} = d_{\max,u}$. The second component, $\mathcal{H}_t = \{\mathcal{H}_{u,t}\}_{u=1}^{n}$, records the team's historical trajectories, with $\mathcal{H}_{u,t}$ being the sequence of task nodes completed by UAV u since departing from $\mathcal{T}_0$, initialized as $\mathcal{H}_{u,0} = (\mathcal{T}_0)$. The final component is a dynamic multi-constraint feasibility mask, $M_t \in \{0,1\}^{n\times(m+1)}$. Concretely, M_t is computed from an element-wise logical AND of several masks: the UAV availability mask ($M_{u,t}$), which zeros out rows for UAVs that have returned to $\mathcal{T}_0$; the dependency mask ($M_{dep,t}$), which ensures task prerequisites are met and prevents reassignment of completed tasks; and the energy and return-to-base mask ($M_{en,t}$). The dependency mask applies only to task nodes. The energy mask confirms a UAV has sufficient range for a round trip, as determined by:

$$M_{\text{en},t}(u,j) = \mathbb{1}[\text{dist}(p_{u,t}, \mathcal{T}_j) + \text{dist}(\mathcal{T}_j, \mathcal{T}_0) \le d_{u,t}] \quad (1)$$

where $\text{dist}(\cdot,\cdot)$ is the Euclidean distance and $\mathbb{1}[\cdot]$ is the indicator function. This mask strictly restricts the action space, guaranteeing that hard constraints like flight distance are never violated during deployment.

b) Action: The action a_t at step t is a joint action that assigns new tasks to all available UAVs. This action is a set of UAV-to-task assignment pairs, denoted as $a_t = \{(u_1, j_1), \dots\} \subseteq U \times \mathcal{T}$. Since the node set $\mathcal{T}$ includes the base station ($\mathcal{T}_0$), an assignment to this node ($j = 0$) represents a return-to-base action. For a valid assignment, each UAV can appear at most once in this set, and each task can appear at most once, except for $\mathcal{T}_0$, which may be assigned to multiple UAVs. Furthermore, every pair (u, j) in a_t must satisfy the feasibility mask, meaning $M_t(u, j) = 1$.

c) Reward: To align the agent's objective with the global optimization goal, we use a sparse terminal reward given only at the end of an episode. No intermediate rewards are provided. When an episode concludes because no feasible UAV–task pair remains, a single total reward $\mathcal{R}$ is calculated. This reward is the sum of the predefined values for all tasks completed during the episode: $\mathcal{R} = \sum_{k\in\mathcal{H}_{final}} r_k$, where $\mathcal{H}_{final}$ is the set of all completed tasks and r_k is the reward for task $\mathcal{T}_k$.

B. Policy network

After formalizing this problem as a MDP, we design a parameterized policy network $\pi_\theta(a|s)$, to learn a mapping from a state s to a joint action a. This network uses a Transformer-based encoder-decoder architecture, as shown in Fig. 2.

a) Causal Extension: To model the long-range causal dependencies between tasks, we design a causal extension module to explicitly capture multi-hop dependency relationships. The standard dependency matrix D only represents direct prerequisite relationships. To address this, we first define an r-hop reachability matrix, $R^{(r)}$, whose elements are given by $(R^{(r)})_{ij} = \mathbb{1}[(D^r)_{ij} > 0]$, where $\mathbb{1}[\cdot]$ is the indicator function that checks for the existence of a dependency path of length r. To isolate the shortest causal distances and create mutually exclusive dependency channels, we further extract adjacency matrices E_r that represent shortest paths of exactly r hops. The recursive definition ensures that $(E_r)_{ij} = 1$ if and only if the shortest dependency path from $\mathcal{T}_i$ to $\mathcal{T}_j$ is exactly r hops:

$$E_1 = D, \quad E_r = R^{(r)} \wedge \left(\neg \bigvee_{l=1}^{r-1} E_l\right) \quad \text{for } r > 1 \quad (2)$$

where $\wedge$, $\neg$, and $\bigvee$ represent element-wise logical operations. For example, in the illustrated "Causal Extension Example," the relation $\mathcal{T}_1 \to \mathcal{T}_3$ is a 1-hop relationship ($\mathcal{K}^1$). Because a path $\mathcal{T}_1 \to \mathcal{T}_3 \to \mathcal{T}_5$ exists and no shorter path is available, $(\mathcal{T}_1, \mathcal{T}_5)$ is classified as a 2-hop relationship ($\mathcal{K}^2$). This process decomposes the dependency graph into multiple, mutually exclusive causal channels. To stabilize information propagation along these channels, we perform row-normalization on each E_r by dividing by the out-degree of each node, $\deg_i^{(r)} = \sum_j (E_r)_{ij}$, to obtain a row-stochastic matrix $\tilde{E}_r$. The normalized matrix $\tilde{E}_r$ is calculated as follows:

$$(\tilde{E}_r)_{ij} = \begin{cases} (E_r)_{ij} / \deg_i^{(r)}, & \text{if } \deg_i^{(r)} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The resulting matrix $\tilde{E}_r$ encodes successor relationships. We also use its transpose $\tilde{E}_r^T$, to model predecessor relationships. Finally, we construct a set of matrices for all these causal

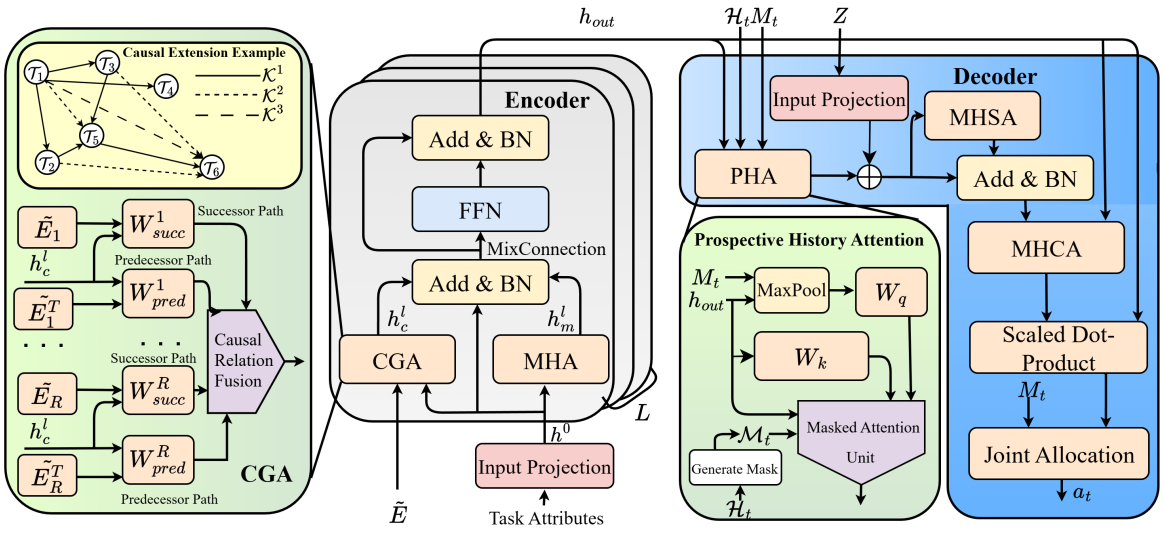


Fig. 2. The overall architecture of C2T-JA. The encoder, featuring a dual-branch Causal-Graph Attention (CGA) and Multi-Head Attention (MHA), computes task embeddings that capture both causal and spatial relationships. The Prospective History Attention (PHA) module generates a context vector by processing the encoder output, the feasibility mask M_t, and the team's trajectory history $\mathcal{H}_t$. Finally, the decoder integrates this context, performs multi-head self-attention (MHSA) and a multi-head cross-attention (MHCA), and executes a joint allocation step constrained by M_t to output a valid joint action a_t.

channels $\tilde{E} = \{\tilde{E}_1, \tilde{E}_1^T, \ldots, \tilde{E}_R, \tilde{E}_R^T\}$, which serves as input to the subsequent modules. Here, R is the maximum number of hops to explore. This entire mechanism provides a strong structural inductive bias, allowing the network to learn a disentangled, hierarchical representation of task precedence rather than treating all dependencies monolithically.

b) Encoder: We design a dual-branch hybrid attention encoder to produce rich embeddings capturing both task attributes and structure. The encoder processes information in two parallel branches: a multi-head self-attention module (MHA) and a Causal-Graph Attention (CGA) module. The raw input consists of two parts: the initial feature embeddings for each task, formed by concatenating its coordinates and reward value, and the set of causal extension matrices $\tilde{E}$. We first project the task attributes into initial task embeddings $h^0 \in \mathbb{R}^{(m+1)\times d_h}$, where the embedding dimension is $d_h = 128$. The encoder is composed of L structurally identical layers, and we use the superscript $l \in \{1, \ldots, L\}$ to denote the layer number. The first branch is a standard MHA layer [27] with 8 heads, which captures feature similarities and spatial correlations. For each layer l, this module takes the task embeddings h^{l-1} as input to produce globally-informed representations $h_m^l = \text{MHA}^{(l)}(h^{l-1})$, where the weights are unique to each layer. The CGA branch is designed to encode causal dependencies and propagate information along the causal channels. This module establishes parallel predecessor–successor paths for each hop count $r \in \{1, \ldots, R\}$. In the predecessor path, input features h^{l-1} are linearly transformed by a dedicated weight matrix $W_{pred}^{r(l)}$, unique for each hop count r, and then aggregated from r-hop predecessors using $\tilde{E}_r^T$ as the propagation operator. The successor path acts analogously using $W_{succ}^{r(l)}$ and $\tilde{E}_r$. This fusion of information from multiple channels is the core of the CGA mechanism. The model learns to dynamically weigh the importance of each of the $2R$ predecessor and successor paths via a learned attention vector, resulting in a unified representation $\hat{h}_c^l$:

$$\hat{h}_c^l = \sum_{r=1}^{R} \left[\alpha'^{(l)}_{r,\text{succ}} \cdot \left(\tilde{E}_r \left(h^{l-1} W_{\text{succ}}^{r(l)} \right) \right) + \alpha'^{(l)}_{r,\text{pred}} \cdot \left(\tilde{E}_r^T \left(h^{l-1} W_{\text{pred}}^{r(l)} \right) \right) \right] \tag{4}$$

$$h_c^l = \hat{h}_c^l W_{co}^l \tag{5}$$

where $\alpha'^{(l)}_{r,pred}$ and $\alpha'^{(l)}_{r,succ}$ are attention weights derived from a learnable, per-layer parameter vector $\alpha^{(l)}$ of length $2R$ via softmax normalization. After obtaining h_m^l and h_c^l, we fuse them using a MixConnection layer. The fusion weights are computed from a learnable, per-layer parameter vector $\beta^{(l)} = [\beta_m, \beta_c]$ by applying a softmax function $\sigma(\cdot)$ to it, forming a learnable convex combination where $\sigma(\beta^{(l)})_m + \sigma(\beta^{(l)})_c = 1$. This dynamically balances feature and structural information. The fused features then pass through a standard transformer block: a residual connection with Batch Normalization (BN), feed-forward network (FFN: Linear–ReLU–Linear, hidden size 512), and a final residual connection with BN. The computational flow is:

$$\hat{h}^l = \sigma(\beta^{(l)})_m h_m^l + \sigma(\beta^{(l)})_c h_c^l \tag{6}$$

$$\tilde{h}^l = \text{BN}^l(h^{l-1} + \hat{h}^l) \tag{7}$$

$$h^l = \text{BN}^l\big(\tilde{h}^l + \text{FFN}^l(\tilde{h}^l)\big) \tag{8}$$

After L layers, the encoder outputs a tensor $h_{out} \in \mathbb{R}^{(m+1)\times d_h}$ containing the final node embeddings.

c) Decoder: The encoder's output h_{out} and the current state s_t serve as the inputs to the decoder. The initial UAV state features Z are linearly projected to obtain initial UAV embeddings $h_u \in \mathbb{R}^{n\times d_h}$. Concurrently, we design a context-aware module named the Prospective History Attention (PHA), which takes h_{out}, $\mathcal{H}_t$, and M_t as input to dynamically extract the most relevant historical information for the current decision. First, we generate a query vector Q for each UAV by applying max-pooling over the embeddings of all its accessible task nodes. This excludes $\mathcal{T}_0$ because its zero reward and fixed coordinates would dilute the query's focus on actual future targets. This query vector is then used as an attention probe to match against key vectors derived from all task nodes, enabling a precise review of past trajectories. To ensure this review is confined to each UAV's own past trajectory, we apply a historical mask $\mathcal{M}_t$ that masks out attention scores for any node not present in that UAV's path history $\mathcal{H}_{u,t}$. The final context vector, $h_{\text{context}} \in \mathbb{R}^{n\times d_h}$, is then obtained by this masked, attention-weighted sum. The computational flow is as follows:

$$Q = W^q_{\text{context}} \cdot \text{MaxPool}_{j\geq 1}\left(M_t\left(h_{out}\right)\right), \tag{9}$$

$$K = W^k_{\text{context}} h_{out}, \quad V = h_{out} \tag{10}$$

$$h_{context} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_h}} + \mathcal{M}_t\right)\cdot V \tag{11}$$

where W^q_{context} and W^k_{context} are learnable weight matrices; and $M_t(h_{out})$ denotes the set of embeddings of all currently available tasks for each UAV.

In the decoding phase, we first fuse the context vector h_{context} with the initial UAV embeddings h_u and process them through a multi-head self-attention (MHSA) module to update UAV representations $\hat{h}_u$, which allows UAVs to reason about each other's states and attended histories. Subsequently, we employ a multi-head cross-attention (MHCA) module, using $\hat{h}_u$ as the Query and the task node embeddings h_{out} as the Key and Value, to further refine the UAV representations. The output, h'_u, captures the complex interactive relationships between each UAV and all tasks. Finally, we compute the final allocation logits matrix, $S \in \mathbb{R}^{n\times(m+1)}$, by taking the scaled dot-product between the UAV representations in h'_u and the task embeddings h_{out}, followed by a $\tanh$ clipping function. The core computation is as follows:

$$\tilde{h}_u = h_u + h_{context}, \quad \hat{h}_u = \text{BN}(\text{MHA}(\tilde{h}_u) + \tilde{h}_u) \tag{12}$$

$$h'_u = \text{MHCA}(\hat{h}_u, h_{out}, h_{out}) \tag{13}$$

$$S_{ij} = C \cdot \tanh\left(\frac{((h'_u)_i)^T (h_{out})_j}{\sqrt{d_h}}\right) \tag{14}$$

where S_{ij} is the compatibility score for assigning node j to UAV i, and $C = 10$ is a clipping hyperparameter. The joint action a_t is constructed through an auto-regressive decoding procedure. At each step, we first apply the mask M_t to the logits matrix S, masking out all invalid UAV-to-task assignments. The remaining valid logits are then flattened into a vector and converted into a probability distribution over all currently possible assignments using a softmax function. Then we select a single assignment (u, j) from this distribution, using either sampling or greedy. After a pair is selected, we update the constraint mask to reflect this assignment. A key special case in this update is that if a task node is assigned, both the UAV and the task are masked out. If $\mathcal{T}_0$ is assigned, only the UAV is masked, while $\mathcal{T}_0$ remains available as a many-to-one destination. This process is repeated until all available UAVs are assigned a task, forming a joint decision that satisfies global constraints.

C. Training

We optimize $\pi_\theta(a|s)$ using REINFORCE with a greedy rollout baseline. For combinatorial routing, this directly compares with the best-found greedy strategy, effectively stabilizing high-variance updates without the complexity of PPO. Our framework pairs a policy network π_θ with a structurally identical baseline network π_b. During each episode, the policy network generates a trajectory Π by sampling, while the baseline network generates a deterministic trajectory Π_b greedily. The difference in their rewards $(R(\Pi) - R(\Pi_b))$ is used as an advantage estimate to compute the policy gradient. The gradient of the final loss function reads:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{\Pi\sim\pi_\theta}\left[(R(\Pi) - R(\Pi_b))\nabla_\theta \log \pi_\theta(\Pi|s_0)\right] \tag{15}$$

The baseline's return $R(\Pi_b)$ is detached from the computation graph to ensure an unbiased gradient. Here, Π represents the full sequence of joint allocations made across all decision steps in an episode. To ensure the baseline remains a strong reference, its weights are periodically updated. When the policy network outperforms the baseline on the validation set (paired t-test, $p < 0.05$), its weights are copied to the baseline network. During training, the average terminal reward steadily converges within the 150 epochs, demonstrating stable policy learning despite the lack of intermediate guidance.

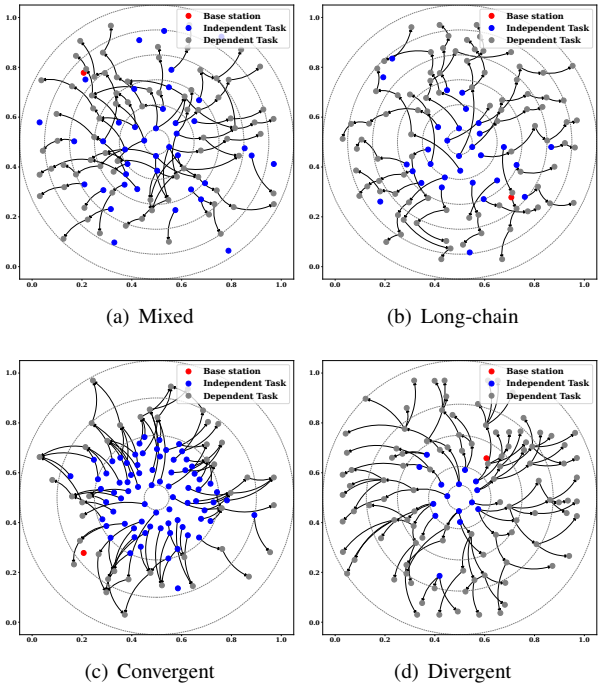
V. Experiments

In this section, we conduct a comprehensive set of experiments to validate our framework's effectiveness. We compare our method against several baselines in terms of solution quality and computation time across a range of scenarios with varying task scales and dependency complexities. To provide a deeper analysis, we also conduct an ablation study to verify the necessity of each core component and run experiments in dynamic scenarios to assess the model's online decision-making capabilities under uncertainty. The code can be found at: https://github.com/zxt6789/C2T-JA.

A. Experimental Setting

To construct the test scenarios, we first deploy task points uniformly in a normalized two-dimensional space, adhering to a concentric ring layout, sector partitioning, and minimum distance constraints. Simultaneously, the drone base station is randomly placed in an outer annular region that does

not overlap with the central task area. Subsequently, dependencies are established between tasks in adjacent regions to programmatically generate four logical structures with distinct challenges: Mixed, Long-Chain, Convergent, and Divergent, as illustrated in Fig. 3. All task reward values are uniformly set to 1.0. In the scale tests, a shorter initial flight distance 1.8 is used to impose stricter energy constraints, while in the structure tests, a longer initial flight distance 2.0 ensures solution feasibility, focusing the evaluation on the algorithm's ability to handle dependency structures.



(a) Mixed (b) Long-chain

(c) Convergent (d) Divergent

Fig. 3. Visualization of the four task dependency graphs used as test scenarios: (a) Mixed, (b) Long-chain, (c) Convergent, and (d) Divergent.

All experiments are conducted on a workstation equipped with an NVIDIA RTX 4090 GPU. The encoder layer count is set to $L = 3$. Since our generated DAGs are predominantly shallow (Fig. 3), we cap the causal channel depth at $R = 3$ to balance modeling coverage against complexity, as deeper channels offered diminishing validation gains. The model is trained for 150 epochs, with each epoch comprising 128,000 dynamically generated instances. Training is performed using mini-batches of size 512. At the end of each epoch, the current policy is evaluated on a fixed validation set (2048 instances), and a paired t-test is used to determine whether it significantly outperforms the baseline network, guiding the decision to update the baseline. For optimization, we employ the Adam optimizer with an initial learning rate of 1×10^{-4}, which undergoes exponential decay with a factor of 0.995 after each training epoch. To further ensure training stability, gradient clipping is applied, limiting the L2 norm of gradients to 3.0. To guarantee experimental reproducibility, the global random seed is uniformly set to 126.

B. Comparison Analysis

We compare our C2T-JA against baselines from three categories: Gurobi as the exact solver, OR-Tools as the conventional heuristic, and AM-joint, a learning-based method adapted from the Attention Model [10] for joint multi-agent task allocation. To ensure a fair comparison, we configured the OR-Tools solver to align with our primary objective: maximizing the number of completed tasks. This deviates from its typical goal of minimizing travel distance. For Gurobi, we formulate a MIP that maximizes the number of completed tasks under the same constraints to ensure objective alignment with all baselines. As an exact solver, Gurobi is computationally intensive; thus, we assess it on 30 test instances per problem size with a 3600-second time limit, whereas the other baselines are evaluated on 512 instances without time constraints. For non-exact baselines and our method, we report wall-clock computation time per instance without an artificial cutoff; this reflects practical decision latency rather than exhaustive search. For both the AM-joint baseline and our C2T-JA, we apply greedy decoding during testing as outlined in the methodology. AM-joint also uses the identical feasibility masks (dependency, energy, and return-to-base) to ensure parity. To evaluate performance across different scales, we constructed test instances with a Mixed dependency structure, involving 80, 100, 150, and 200 tasks, each tested with teams of 4 and 6 UAVs. Instances are denoted as T[tasks]U[UAVs] (e.g., T80U4 for 80 tasks and 4 UAVs). The results are presented in Table I, which summarizes the average task completion rate (ATCR), computation time, and performance gap for all methods. To quantify the relative performance, we define the gap against the best-performing approach as:

$$\text{Gap} = \frac{\text{ATCR}_{\text{best}} - \text{ATCR}}{\text{ATCR}_{\text{best}}} \times 100\%$$

where $\text{ATCR}_{\text{best}}$ is the highest ATCR achieved by any method for that specific test case.

Gurobi's performance as an exact solver highlights the problem's computational complexity. While achieving a 61.10% ATCR on the T80U4 instance, its solution quality sharply deteriorates with scale, dropping to a mere 3.00% on T200U6 and resulting in a performance gap of 95.39% against the best method, all while consistently hitting the 3600-second time limit. This underscores the severe scalability limitations of exact methods for this problem class. In contrast, learning-based and heuristic methods are far more efficient. However, a clear performance gap emerges between them. The AM-joint baseline, while extremely fast, consistently underperforms, exhibiting a performance gap relative to the best solution that ranges from 4.11% to 10.13%. This suggests its standard attention mechanism is insufficient for navigating the complex constraints.

The most revealing comparison is with OR-Tools. While OR-Tools achieves the highest ATCR on the smallest T80 instances, a critical trend emerges as the problem scales. Starting from the T100U4 instance, our C2T-JA model begins to outperform OR-Tools, achieving a 62.96% ATCR compared to 60.66%. This advantage is sustained across all larger instances. For example, on the largest T200U6 instance, C2T-JA achieves an ATCR of 65.01% versus OR-Tools'

TABLE I
PERFORMANCE COMPARISON ACROSS DIFFERENT SCALES.

Instance	Method	ATCR(%)	Time (s)	Gap(%)
T80U4	Gurobi	61.100 ± 2.3100	3600	15.07
	OR-Tools	**71.940** ± 0.0049	7.12240	**0.00**
	AM-joint	64.658 ± 0.4530	**0.00715**	10.12
	C2T-JA(ours)	68.250 ± 0.3990	0.01037	5.13
T80U6	Gurobi	73.880 ± 1.2600	3600	15.66
	OR-Tools	**87.600** ± 0.0052	11.38760	**0.00**
	AM-joint	80.208 ± 0.5780	**0.00740**	8.44
	C2T-JA(ours)	82.595 ± 0.5340	0.00989	5.71
T100U4	Gurobi	42.480 ± 2.3800	3600	32.53
	OR-Tools	60.660 ± 0.0046	7.80540	3.65
	AM-joint	59.002 ± 0.3610	**0.00959**	6.29
	C2T-JA(ours)	**62.960** ± 0.3750	0.01184	**0.00**
T100U6	Gurobi	42.280 ± 3.7400	3600	47.87
	OR-Tools	78.400 ± 0.0051	12.17000	3.33
	AM-joint	76.025 ± 0.5150	**0.00955**	6.26
	C2T-JA(ours)	**81.104** ± 0.4704	0.01108	**0.00**
T150U4	Gurobi	35.880 ± 0.6400	3600	35.13
	OR-Tools	52.840 ± 0.0038	30.87600	4.47
	AM-joint	51.320 ± 0.3190	**0.01345**	7.22
	C2T-JA(ours)	**55.314** ± 0.3160	0.02096	**0.00**
T150U6	Gurobi	37.560 ± 3.7900	3600	47.75
	OR-Tools	69.320 ± 0.0048	45.11690	3.57
	AM-joint	68.022 ± 0.4300	**0.01361**	5.38
	C2T-JA(ours)	**71.889** ± 0.4160	0.02222	**0.00**
T200U4	Gurobi	16.770 ± 1.7700	3600	65.66
	OR-Tools	45.680 ± 0.0039	67.17160	6.46
	AM-joint	44.089 ± 0.2880	**0.01816**	9.72
	C2T-JA(ours)	**48.837** ± 0.2980	0.03308	**0.00**
T200U6	Gurobi	3.000 ± 0.0000	3600	95.39
	OR-Tools	62.420 ± 0.0046	109.88200	3.98
	AM-joint	62.335 ± 0.3940	**0.01769**	4.11
	C2T-JA(ours)	**65.007** ± 0.3770	0.03458	**0.00**

62.42%. Crucially, this higher performance is achieved with unparalleled speed. On the same T200U6 instance, C2T-JA's decision-making is over three orders of magnitude faster than OR-Tools (0.035 s versus 109.882 s). This combination of higher solution quality on large-scale instances and near-instantaneous decision-making showcases the key advantage of our end-to-end approach. By learning a direct policy from state to joint action, C2T-JA bypasses the costly combinatorial search that limits traditional methods, achieving an exceptional balance of performance and efficiency.

Additionally, to assess the algorithm's ability to handle different dependency structures, we conduct tests across the four logical structures at a fixed problem scale (T120U4). This controlled setting is chosen for the ablation study to isolate the model's structural reasoning capabilities from the confounding variable of problem scale.

Table II confirms the necessity of our core designs: removing either the causal extension (w/o CE) or the dual-branch encoder (w/o CGA) causes significant performance drops across all structures. In Convergent cases, explicit multi-hop dependency modeling (CGA+CE) together with context-aware decoding (PHA) helps prioritize bottleneck paths and maintain temporal consistency across agents (w/o PHA 6.85%, w/o CGA 5.03%). In Divergent cases, once a key prerequisite is met, deep causal modeling (CGA/CE: 6.75%/4.05%) and the joint-allocation decoder enable one-shot, globally coordinated assignment, improving parallel

TABLE II
ABLATION STUDY AND PERFORMANCE ON DIFFERENT DEPENDENCY STRUCTURES.

Instance	Method	ATCR(%)	Time (s)	Gap(%)
Mixed	Gurobi	41.010 ± 0.7500	3600	37.34
	OR-Tools	63.850 ± 0.0042	19.7620	2.44
	AM-joint	63.013 ± 0.3600	**0.0109**	3.72
	C2T-JA(ours)	**65.448** ± 0.3600	0.0167	**0.00**
	w/o CGA	62.388 ± 0.4210	0.0148	4.68
	w/o PHA	63.605 ± 0.3450	0.0146	2.82
	w/o CE	63.675 ± 0.3300	0.0151	2.71
Divergent	Gurobi	31.810 ± 2.4300	3600	50.65
	OR-Tools	64.200 ± 0.0042	20.8027	0.40
	AM-joint	60.763 ± 0.3310	**0.0109**	5.73
	C2T-JA(ours)	**64.458** ± 0.3390	0.0171	**0.00**
	w/o CGA	60.106 ± 0.3810	0.0154	6.75
	w/o PHA	63.791 ± 0.3490	0.0144	1.03
	w/o CE	61.847 ± 0.3200	0.0163	4.05
Convergent	Gurobi	43.960 ± 4.1900	3600	45.54
	OR-Tools	79.690 ± 0.0028	25.2977	1.28
	AM-joint	76.810 ± 0.3770	**0.0099**	4.85
	C2T-JA(ours)	**80.721** ± 0.3500	0.0166	**0.00**
	w/o CGA	76.663 ± 0.3860	0.0149	5.03
	w/o PHA	75.190 ± 0.4930	0.0141	6.85
	w/o CE	78.663 ± 0.5040	0.0153	2.55
Long-Chain	Gurobi	36.160 ± 2.1700	3600	43.44
	OR-Tools	62.710 ± 0.0041	19.312	1.91
	AM-joint	61.463 ± 0.3690	**0.0099**	3.86
	C2T-JA(ours)	**63.929** ± 0.3500	0.0166	**0.00**
	w/o CGA	61.901 ± 0.4100	0.0144	3.17
	w/o PHA	61.219 ± 0.3590	0.0143	4.24
	w/o CE	62.557 ± 0.3440	0.0152	2.15

rollout over sequential heuristics. In Long-Chain cases, explicit long-range reasoning (CGA/CE) and history-guided decoding (PHA 4.24%, CE 2.15%) sustain long-horizon planning and mitigate proximity bias. In Mixed cases, which blend these patterns, we observe cumulative benefits from all modules and consistent degradations when any is removed. Overall, these results indicate that C2T-JA synergistically reasons over the causal task structure and the team state; removing any core module weakens structural foresight or coordination and leads to measurable performance loss.

C. Experiments in Dynamic Scenarios

To evaluate the robustness of C2T-JA, we test its online decision-making in dynamic scenarios with partial observability, using the T150U6 instance with a Mixed dependency structure as the testbed. Here, each UAV can only detect tasks within a limited perception radius, simulating real-world informational constraints. By systematically varying this radius and comparing performance against the static, full-information case, we quantify the model's resilience to environmental uncertainty.

TABLE III
PERFORMANCE UNDER PARTIAL OBSERVABILITY.

Instance	radius	ATCR(%)	Time (s)	Gap(%)
T150U6	Static	**71.889** ± 0.416	**0.0222**	**0.00**
	0.5	71.223 ± 0.419	0.1150	0.93
	0.4	70.014 ± 0.535	0.1224	2.61
	0.3	60.921 ± 1.310	0.1264	15.26
	0.2	38.012 ± 1.699	0.1390	47.12

The results in Table III confirm the model's sensitivity to observability. While the performance gap is minimal

(0.93%) with a near-global perception radius of 0.5, it increases sharply to 47.12% as the radius shrinks to 0.2. This degradation is an expected consequence of our policy's training on full information, which makes it adept at global optimization but not inherently robust to the information scarcity required for effective long-horizon planning. Without sufficient perceptual range, the model fails to anticipate conflicts and opportunities among spatially dispersed tasks, leading to myopic decisions that are only locally optimal. Nonetheless, the policy maintains its core advantage of near-instantaneous decision-making (< 0.14 s), revealing a key trade-off between global optimality and computational speed. This suggests its value where rapid replanning is critical. Future work could improve robustness by incorporating partial observability during training or integrating local search algorithms.

VI. CONCLUSION

In this paper, we addressed the challenging problem of multi-UAV task allocation under complex causal dependencies and energy constraints by proposing C2T-JA. Our framework explicitly models multi-hop dependencies to generate globally coordinated joint actions for the entire UAV team. Comprehensive experiments demonstrated that C2T-JA significantly outperforms established baselines, particularly on large-scale instances, achieving higher task completion rates while reducing decision times by orders of magnitude. Furthermore, ablation studies confirmed the contribution of our core architectural components, and tests in dynamic scenarios highlighted the policy's robustness.

Future work will explore extending our discrete-time framework to a continuous-time model for handling asynchronous events like stochastic task arrivals. We also plan to investigate multi-objective optimization and bridge the sim-to-real gap by deploying our policies on physical multi-robot systems.

REFERENCES

[1] S. A. Ghauri, M. Sarfraz, R. A. Qamar, M. F. Sohail, and S. A. Khan, "A Review of Multi-UAV Task Allocation Algorithms for a Search and Rescue Scenario," *Journal of Sensor and Actuator Networks*, vol. 13, no. 5, Art. no. 47, 2024.

[2] A. Rejeb, A. Abdollahi, K. Rejeb, and H. Treiblmaier, "Drones in agriculture: A review and bibliometric analysis," *Computers and Electronics in Agriculture*, vol. 198, Art. no. 107017, 2022.

[3] N. H. Motlagh, P. Kortoçi, X. Su, L. Lovén, H. K. Hoel, S. B. Haugsvær, V. Srivastava, C. F. Gulbrandsen, P. Nurmi, and S. Tarkoma, "Unmanned Aerial Vehicles for Air Pollution Monitoring: A Survey," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21687–21704, 2023.

[4] M. Gini, "Multi-Robot Allocation of Tasks with Temporal and Ordering Constraints," in *Proc. 31st AAAI Conf. on Artificial Intelligence (AAAI)*, pp. 4863–4869, 2017.

[5] M. Lippi and A. Marino, "A Mixed-Integer Linear Programming Formulation for Human Multi-Robot Task Allocation," in *Proc. 30th IEEE Int. Conf. on Robot & Human Interactive Communication (RO-MAN)*, pp. 1017–1023, 2021.

[6] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast Scheduling of Robot Teams Performing Tasks With Temporospatial Constraints," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.

[7] E. Bischoff, F. Meyer, J. Inga, and S. Hohmann, "Multi-Robot Task Allocation and Scheduling Considering Cooperative Tasks and Precedence Constraints," in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC)*, pp. 3949–3956, 2020.

[8] 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.

[9] X. Bai, A. Fielbaum, M. Kronmuller, L. Knödler, and J. Alonso-Mora, "Group-Based Distributed Auction Algorithms for Multi-Robot Task Assignment," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1292–1303, 2023.

[10] W. Kool, H. van Hoof, and M. Welling, "Attention, Learn to Solve Routing Problems!," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[11] H. Mao, Z. Zhang, Z. Xiao, and Z. Gong, "Modelling the Dynamic Joint Policy of Teammates with Attention Multi-Agent DDPG," in *Proc. 18th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1108–1116, 2019.

[12] Google OR-Tools, "Vehicle Routing," *OR-Tools Documentation*. [Online]. Available: `https://developers.google.com/optimization/routing/`. Accessed: 2026-3-6.

[13] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, Version 13.0. [Online]. Available: `https://docs.gurobi.com/projects/optimizer/en/current/`. Accessed: 2026-3-6.

[14] D. Bredström and M. Rönnqvist, "Combined Vehicle Routing and Scheduling with Temporal Precedence and Synchronization Constraints," *European Journal of Operational Research*, vol. 191, no. 1, pp. 19–31, 2008.

[15] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, "xBots: An Approach to Generating and Executing Optimal Multi-Robot Plans with Cross-Schedule Dependencies," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 115–122, 2012.

[16] E. G. Jones, M. B. Dias, and A. Stentz, "Time-Extended Multi-Robot Coordination for Domains with Intra-Path Constraints," *Autonomous Robots*, vol. 30, no. 1, pp. 41–56, 2011.

[17] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot task assignment with deadlines for tasks," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 3007–3013, 2013.

[18] M. McIntire, E. Nunes, and M. Gini, "Iterated Multi-Robot Auctions for Precedence-Constrained Task Scheduling," in *Proc. 2016 Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1078–1086, 2016.

[19] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[21] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing," in *Proc. 36th Int. Conf. on Machine Learning (ICML)*, vol. 97, pp. 21–29, 2019.

[22] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling Relational Data with Graph Convolutional Networks," in *Proc. European Semantic Web Conf. (ESWC)*, LNCS 10843, pp. 593–607, 2018.

[23] D. Chen, L. O'Bray, and K. Borgwardt, "Structure-Aware Transformer for Graph Representation Learning," in *Proc. 39th Int. Conf. on Machine Learning (ICML)*, vol. 162, pp. 3469–3489, 2022.

[24] S. Paul, P. Ghassemi, and S. Chowdhury, "Learning Scalable Policies over Graphs for Multi-Robot Task Allocation using Capsule Attention Networks," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 8815–8822, 2022.

[25] G. Mialon, D. Chen, M. Selosse, and J. Mairal, "GraphiT: Encoding Graph Structure in Transformers," arXiv preprint arXiv:2106.05667, 2021.

[26] S. Iqbal and F. Sha, "Actor-Attention-Critic for Multi-Agent Reinforcement Learning," in *Proc. 36th Int. Conf. on Machine Learning (ICML)*, vol. 97, pp. 2961–2970, 2019.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 5998–6008, 2017.