IAOM: An Intention-Aware Optimization Model With GNN-Based Policy Learning for Dynamic Multi-Robot Cooperative Task Allocation

Qingyang Long and Liping Liang

***Abstract*—This paper studies dynamic multi-robot cooperative task allocation in multi-stage scout-rescue missions with unknown and evolving task demands. We propose IAOM, an Intention-Aware Optimization Model that encodes each robotś decision state–target task, role, and planner-derived ETA–as an intention vector and organizes robots and tasks into an execution-aware intention graph capturing residual demand and active commitments. A GAT-PPO policy is learned on this graph and coupled to a hybrid A∗ based motion layer, so that allocation, coalition formation, and congestion handling are optimized within a single framework. We evaluate IAOM on grid based urban rescue benchmarks with multiple map sizes and resource density regimes against optimization-, heuristic-, and learning-based baselines. Under abundant and normal resources, IAOM consistently reduces mission makespan and average completion time relative to the best baseline, achieving up to about 20% shorter makespan and about 50% lower completion time with comparable load balance. Under tight resources where robots barely meet total demand, gains are smaller and all methods are largely bounded by resource scarcity, but IAOM remains competitive. This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments.**

***Index Terms*—Multi-robot systems, graph neural networks, multi-robot task allocation, reinforcement learning.**

I. Introduction

Urban fire disasters are becoming more frequent and harder to manage as cities grow denser and environmental conditions become more volatile. In such time-critical and hazardous environments, multi-robot teams are attractive: they can scout large buildings, form ad-hoc coalitions for rescue, and operate safely in smoke-filled or structurally unstable areas [1], [2]. Turning this potential into practice, however, requires more than just assigning robots to static tasks—it demands *dynamic* multi-robot task allocation (MRTA) under progressively revealed task requirements in multi-stage scout-rescue workflows.

Classical MRTA schemes [3] based on optimization or market mechanisms [4], [5], [6] provide strong foundations, but they typically assume known task requirements and near-static task sets, or rely on repeated global re-solving that becomes expensive and brittle in congested layouts. More recent dynamic and rescue-oriented frameworks, such as multi-stage MRTA for scouting and rescue [7] or heterogeneous RL-based schedulers [8], improve robustness to task and capability uncertainty, yet still reason on abstract task graphs and couple motion only through aggregate travel costs. In parallel, learning-based and GNN-based allocators [9], [10], [11], [12], [13] have shown that graph representations and deep policies can scale to large teams, but most of these approaches build on static or slowly varying graphs, with edges parameterized by surrogate quantities such as Euclidean distance or static priority, and treat motion planning as an external module. Cooperative task capacity, synchronized arrivals, and spatial congestion are seldom modeled as first-class graph attributes, which can cause allocation policies that appear optimal under abstract costs to misalign with the actual execution dynamics of grid-based rescue missions.

To address this gap, we propose IAOM, an Intention-Aware Optimization Model for dynamic MRTA in multi-stage scout-rescue missions. The key idea is to treat each robot's planned task, role, and expected arrival time as an explicit intention and let coordination operate directly in this intention space rather than on tasks alone. Unlike prior MRTA formulations that operate primarily on task-centric graphs or cost metrics, IAOM explicitly models robot intentions and systematically couples them with coalition formation and execution-aware learning in a unified framework.

Concretely, IAOM assigns each robot an *intention vector* and lifts all robots and tasks into a directed *intention graph*, whose nodes encode spatial context and residual (initially unknown) task demand, and whose edges represent active robot-task commitments. A GAT [14]-PPO [15] policy then runs on this graph to refine intentions, while a feasibility- and load-aware decoding scheme, coupled with a shared A∗ planner, turns them into capacity-consistent assignments that account for corridor congestion.

The main contributions of this work are threefold:

Intention-aware MRTA representation: We introduce an intention-vector and intention-graph representation that explicitly brings robot intentions into the MRTA loop, enabling coalition formation and reallocation in multi-stage scout-rescue missions with initially unknown task demands.

Received 1 April 2026; accepted 1 June 2026. Date of publication 12 June 2026; date of current version 19 June 2026. This article was recommended for publication by Associate Editor V. Edwards and Editor M. Ani Hsieh upon evaluation of the reviewers' comments. *(Corresponding author: Liping Liang.)*

The authors are with the School of Integrated Circuits, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: raelcc@bupt.edu.cn; liangliping@bupt.edu.cn).

Digital Object Identifier 10.1109/LRA.2026.3703220

2377-3766 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.

Execution-coupled graph policy optimization: We develop a GAT-PPO scheduler that operates directly on the intention graph and is tightly coupled to a path planner via planner-derived ETAs and residual task demand, so that allocation, cooperation, and congestion are optimized within a single execution-aware learning framework.

Extensive evaluation against diverse baselines: We systematically evaluate IAOM on grid-based urban rescue benchmarks with varying map sizes and resource densities, comparing against optimization-based, greedy, team-oriented heuristic, and learning-based GNN schedulers. IAOM consistently improves mission makespan and average completion time over the best baselines while maintaining competitive robot load balance in dynamic cooperative rescue scenarios.

The remainder of this paper is structured as follows: Section II reviews related work. Section III presents the IAOM framework in detail. Section IV provides experimental evaluations. Section V concludes the paper and discusses future work.

II. Related Work

A. Dynamic Multi-Robot Task Allocation

Dynamic Multi-Robot Task Allocation (MRTA) concerns real-time assignment of tasks to robot teams under uncertainty, online task arrival, and cooperative execution constraints. Gerkey and Matarić [16] provided a widely adopted taxonomy along three axes (ST/MT, SR/MR, IA/TA), which we follow by focusing on the ST–MR–IA setting with cooperative tasks. Classical market- and optimization-based methods [4], [5], [6] are effective for static or slowly changing task sets, but their reliance on global re-solving or greedy myopic decisions makes them brittle in crowded, highly dynamic environments.

Recent work has moved toward adaptive and resilient MRTA. SCoBA [17] models stochastic completion times in a hierarchical allocator, Emam et al. [18] update latent capabilities for heterogeneous teams online, and SMT/MIP-based schemes [19], [20] interleave allocation and planning with correctness guarantees. In search-and-rescue, MSMRTA-TOH [7] integrates scouting, heterogeneous capabilities and team formation in a multi-stage assignment pipeline. These methods improve robustness to task and capability uncertainty, but allocation is still computed on abstract task graphs, and motion-level effects such as corridor congestion and synchronized arrivals are only coarsely reflected through cost terms [21]. As a result, there remains a gap between high-level allocation decisions and the fine-grained execution dynamics that actually determine mission completion.

B. Learning-Based and GNN-Based Allocation

Recent work increasingly treats MRTA as a learning problem to avoid hand-crafted heuristics and the scaling limits of exact solvers. A common pattern is to encode robots and tasks as nodes in a graph and use GNNs to produce assignments. ScheduleNet [9] learns to imitate Hungarian-style schedules on bipartite robot-task graphs, Hetero-RL [8] trains attention-based policies for heterogeneous coalition formation via multi-agent RL, and several other GNN-based allocators [10], [11], [12], [13] show that graph representations can improve scalability and generalization.

Across these methods, the interaction graph is typically static or slowly varying; edges are parameterized by surrogate quantities such as Euclidean distance, task priority or skill compatibility; and motion planning is treated as an external cost. ScheduleNet [9] focuses on reproducing expert matchings without explicit modeling of cooperative task capacity or corridor congestion, while Hetero-RL [8] targets heterogeneous capability matching rather than synchronized arrivals on a shared grid. Domain-specific designs for agriculture or industrial scheduling [12], [13] similarly emphasize workload or precedence constraints, leaving spatial congestion and multi-robot coordination largely to the planner or to hand-crafted rewards. This can lead to allocation policies that look near-optimal under abstract cost surrogates but misalign with actual execution dynamics in cluttered environments.

IAOM aims to narrow this gap by operating on a dynamically evolving *intention graph* that is directly coupled to planner-derived arrival times and task capacities. This execution-aware graph lets the policy reason about cooperation and congestion together with allocation, rather than treating motion planning as a separate post-processing step.

C. Intention Modeling in Multi-Agent Coordination

Intention modeling has been widely studied in multi-agent navigation and interaction as a means to anticipate others' behavior. Intent-aware planners infer latent goals of surrounding agents and use them to generate collision-avoiding motions in shared spaces, improving safety and smoothness over purely reactive schemes [22], [23]. In multi-agent reinforcement learning, Kim et al. [24] propose an intention-sharing protocol where agents broadcast compact encodings of imagined future trajectories, while Wu et al. [25] introduce spatial intention maps that render each agent's intent onto an overhead grid for decentralized manipulation.

These approaches primarily use intention for local trajectory prediction, collision avoidance, or behavior inference, and typically keep intention signals separate from cooperative capacity constraints, coalition formation, and MRTA objectives. In particular, intentions inform low-level motion or prediction, but are not used as the main decision space for task allocation. In this work, we bring explicit intention modeling into dynamic ST–MR–IA scheduling by treating intention vectors and an intention graph as the core state for coalition and capacity-aware assignment, and coupling them with GNN-based policy learning and planner-informed rewards so that intention participates in the full allocation-execution loop rather than only in low-level motion planning.

III. Method

This section first formalizes the cooperative search-and-rescue MRTA problem in a grid world, and then presents the three building blocks of IAOM: *Intention Perception*, *Intention Optimization*, and *Hybrid Path Planning and Execution*. Together they form a closed-loop "Perception - Optimization

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.

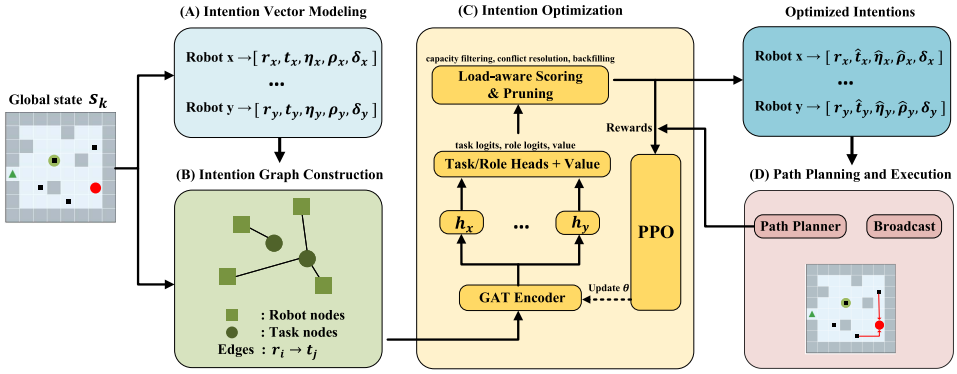


Fig. 1. Block diagram of IAOM. (A) Intention Vector Modeling encodes the global state into per-robot intention vectors with planner-derived ETAs. (B) Intention Graph Construction builds a mixed robot–task intention graph with residual task demand and edges for active commitments. (C) Intention Optimization applies an edge-aware GAT whose task/role/value heads produce action distributions; masked task scores are combined with load-aware scoring and a centralized assignment module to obtain feasible robot–task allocations (trained with PPO). (D) Path Planning and Execution converts optimized intentions into A∗-based paths and dispatches them for decentralized execution. Together, (A)-(D) implement a closed-loop Perception→ Optimization→ Execution cycle.

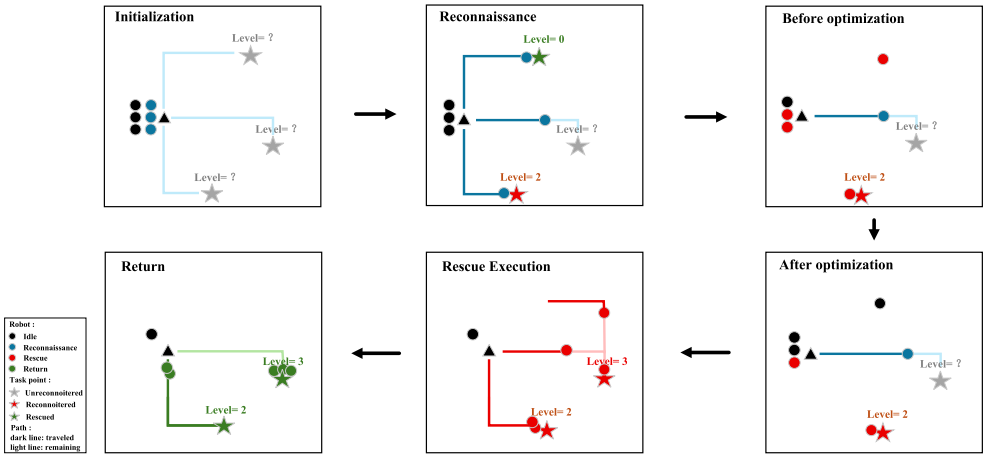


Fig. 2. Illustration of a search and rescue process with 6 robots and 3 task points. Circles denote robots, stars denote tasks, triangles mark entrances, and shaded lines indicate planned paths. Task levels are unknown until reconnoitered; once revealed, idle robots may respond. A circle matching a star's color shows a robot currently targeting that task; blue circles indicate idle robots. Tasks 2 and 3 have equal distances from the entrance, and their reconnaissance robots arrive simultaneously. Tasks with level 0 are discarded upon detection.

- Execution" cycle that realizes global task scheduling and multi-robot coordination on the intention graph. The overall framework is illustrated in Fig. 1.

A. Problem Formulation

We consider a centralized multi-robot coordination problem for search-and-rescue missions in a known grid-based environment. The environment is specified as a 2D occupancy string and parsed into a grid graph $\mathbf{M} = (\mathscr{C}, \mathscr{E}_M)$, where each free cell $\mathbf{c} \in \mathscr{C}$ is four-connected to its non-obstructed neighbors. A team of robots $\mathscr{R} = \{r_1, \ldots, r_N\}$ starts from a common entrance cell $c_{\text{in}} \in \mathscr{C}$ and moves synchronously on the grid using four primitive actions (up, down, left, right). An A∗ planner provides shortest paths between cells and corresponding travel times τ_{ij} from robot r_i to task t_j. The mission evolves in discrete decision steps $k = 0, 1, \ldots, k_{\max}$. A typical scout-rescue process is illustrated in Fig. 2.

A task set $\mathscr{T} = \{t_1, \ldots, t_M\}$ is initialized on accessible cells at mission start and remains fixed; no new tasks are introduced during execution. Each task represents a potential victim site with an initially unknown rescue requirement L_j. Unlike online-arrival MRTA settings, the dynamics considered here arise from progressive revelation of task requirements rather than from the appearance of new tasks. Task t_j follows a three-phase life cycle:

- unreconnoitered: location known, requirement L_j unknown;
- pending: a reconnaissance robot reaches t_j, revealing L_j and activating the task;
- rescued: once L_j rescuers have assembled, participating robots return to c_{in}.

This modeling reflects structured indoor rescue scenarios, where potential victim locations can often be enumerated from prior floorplans or rapid mapping, while the required team effort becomes known only after on-site assessment under smoke or clutter [1], [2]. Although the task set and environment are

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.

static, the coordination problem evolves as reconnaissance reveals demand, inducing time-varying coalition formation and reallocation.

Each robot operates under a discrete role $\rho_i^k \in \{0,1,2,3\}$:

$$\rho_i^k = \begin{cases} 0, & \text{reconnaissance mode,} \\ 1, & \text{rescue mode (en route or executing),} \\ 2, & \text{return mode (evacuating to base),} \\ 3, & \text{idle or waiting state,} \end{cases} \tag{1}$$

and its assignment is indicated by t_i^k ($t_i^k = t_M + 1$ if idle).

At each decision step k, the centralized coordinator maintains the global system state

$$\mathbf{s}_k = \left\{ (\mathbf{p}_i^k,\, \mathbf{z}_i^k)_{i=1}^N,\ (\mathbf{q}_j^k,\, L_j^k,\, \text{status}(t_j))_{j=1}^M \right\}, \tag{2}$$

where $\mathbf{p}_i^k$ represents the position and role of robot r_i, $\mathbf{z}_i^k$ is its current intention vector, $\mathbf{q}_j^k$ is the position of task t_j, L_j^k is its revealed rescue requirement (if known), and status(t_j) encodes the task's life-cycle phase. The state $\mathbf{s}_k$ therefore aggregates both robot-level and task-level information that drive each coordination round.

Whenever idle robots and unrescued tasks coexist, the coordinator triggers a new round consisting of:

- Intention Perception: generate intention vectors and the corresponding intention graph from $\mathbf{s}_k$;
- Intention Optimization: employ a GAT-PPO hybrid policy to update robot intentions on this graph;
- Execution: disseminate optimized intention vectors $\mathbf{z}_i^{k+1}$ and move robots accordingly, updating positions, task states, and roles.

This cycle repeats until all tasks are rescued or the mission horizon $k_{\max}$ is reached.

The objective is to minimize the overall mission makespan, defined as the completion time of the last rescued task:

$$\min_{\pi_\theta}\ J_{\text{makespan}} = \min_{\pi_\theta}\ \mathbb{E}_{\pi_\theta}\left[\max_{t_j \in \mathcal{T}} T_j^{\text{end}} \right], \tag{3}$$

where T_j^{end} denotes the completion time of task t_j. This criterion implicitly captures travel delays, waiting times, and coordination efficiency over the entire mission.

B. Intention Perception

Following the problem formulation, each coordination round starts with an *Intention Perception* phase that maps the global state $\mathbf{s}_k$ into a compact, learnable representation. Given $\mathbf{s}_k$, the coordinator encodes each robot's local decision state as an intention vector and organizes all robots and tasks into an intention graph. This two-level representation (vector + graph) is the interface between environment dynamics and policy learning.

Intention Vector Modeling: Each robot's decision state—where to act, which role to perform, and whether it is currently committed—is represented as $\mathbf{z}_i = [\, r_i,\, t_i,\, \eta_i,\, \rho_i,\, \delta_i \,]$, where r_i is the robot index, t_i the assigned task, η_i the A$*$-based estimated arrival time (ETA) to t_i, $\rho_i \in \{0,1,2,3\}$ the role, and $\delta_i \in \{0,1\}$ a binary flag indicating whether r_i is actively committed in the current round. At the beginning of each coordination round, a lightweight global initializer (by default a random heuristic) uses the current robot/task state to assign feasible rescue targets to truly idle robots and set $(t_i, \eta_i, \rho_i, \delta_i)$ accordingly, while non-idle robots retain their ongoing intentions. The tuple $\mathbf{z}_i$ is not fed to the network directly; instead, its components are dispersed into node and edge features of the intention graph. The formulation is extensible: additional scalars (e.g., energy margin, local congestion) can be appended without changing the downstream pipeline.

Intention Graph Construction: From $\{\mathbf{z}_i\}$ and the current task state, we build a directed *intention graph* $\mathcal{G}_k = (\mathcal{V}, \mathcal{E})$ that summarizes spatial layout, supply–demand balance, and active commitments in round k.

Node representation. The node set $\mathcal{V}$ contains both robot and task nodes. Rather than expanding a task into L_j demand-specific nodes, each task t_j is represented by a single node with residual demand

$$d_j = (L_j - c_j)_+, \qquad c_j = \#\{\, i \mid \delta_i = 1,\ t_i = j \,\}, \tag{4}$$

where $(\cdot)_+$ denotes the non-negative part. Consider a grid map of size $W \times H$ cells. All nodes share a spatial–context prefix $(\hat{x}, \hat{y}, \hat{w}, \hat{h}, \rho_{\text{map}}, \sigma_{\text{supply}}, \hat{A})$, where (x, y) is the grid cell of the robot or task, $\hat{x} = x/W$ and $\hat{y} = y/H$ are normalized coordinates, $\hat{w} = W/64$ and $\hat{h} = H/64$ encode relative map size, ρ_{map} is the current active-task density, σ_{supply} is a global robots-to-demand coefficient indicating whether the system is operating under resource surplus or scarcity, and $\hat{A} = (WH)/(64^2)$ is a normalized area term. On top of this prefix we add type-specific semantics: for a robot node r_i, binary indicators such as *idle* $(0/1)$ and *active* $(0/1$, true when $\delta_i = 1)$; for a task node t_j, raw integer fields L_j and d_j, a truncated waiting feature $waiting_norm = \min(\text{waiting_time}/T_{\max}, 1)$, and a *pending* flag $(0/1)$. Each node feature vector hence takes the form $\mathbf{x}_v = [\, \hat{x}, \hat{y}, \hat{w}, \hat{h}, \rho_{\text{map}}, \sigma_{\text{supply}}, \hat{A},\ \text{type-specific bits/scalars} \,]$, with discrete states encoded as $0/1$ and integer quantities (e.g., L_j, d_j) kept in their raw form.

Edge formation. Directed edges capture *active* robot–task commitments. For each intention $\mathbf{z}_i = [r_i, t_j, \eta_i, \rho_i, \delta_i]$ we add $(r_i \to t_j) \in \mathcal{E} \iff \delta_i = 1$. Each such edge carries an attribute vector

$$\mathbf{e}_{r_i \to t_j} = [\eta_i^+,\ \rho_i,\ L_j,\ \text{pending}_j], \qquad \eta_i^+ = \max(\eta_i, 0), \tag{5}$$

where η_i^+ is the non-negative ETA, ρ_i is the role code, L_j the task's total rescue level, and $\text{pending}_j \in \{0,1\}$ indicates whether t_j is active but not yet completed.

We deliberately add edges only for robot–task pairs with $\delta_i = 1$ so that the graph remains sparse and focused on current commitments, avoiding the $O(|\mathcal{R}||\mathcal{T}|)$ cost and noise of a fully connected bipartite graph. Global context (e.g., $\rho_{\text{map}}, \sigma_{\text{supply}}, d_j$) is embedded in all node features and self-loops are used in the GAT layers, so even idle robots receive updates reflecting the overall task landscape while message passing along active edges concentrates capacity on tasks already attracting commitments.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.

This yields a compact, differentiable intention graph that encodes spatial layout, supply–demand tension, and commitment structure as the input to IAOM's intention optimization.

C. Intention Optimization

Given the intention graph $\mathscr{G}_k$, the policy refines intentions only for robots requiring coordination, while task nodes act as semantic anchors providing feasible slots, residual demand, and urgency cues.

Graph-based Encoding: A two-layer edge-aware Graph Attention Network (GAT) is applied to $\mathscr{G}_k$ to obtain structure-sensitive node embeddings. For a node v with neighbors $u \in \mathcal{N}(v)$, the attention mechanism computes

$$\alpha_{uv} = \mathrm{softmax}_u\big(\phi(W\mathbf{x}_u, W\mathbf{x}_v, \mathbf{e}_{uv})\big), \tag{6}$$

where $\phi(\cdot)$ is an edge-aware GATConv operator that uses node features and edge attributes (ETA/role/task-state). Self-loops are added for stable aggregation. For each robot node, its embedding $\mathbf{h}_i$ is decoded by three independent linear projection heads:

$$\boldsymbol{\ell}_i^{\text{task}} = W_t\mathbf{h}_i, \quad \boldsymbol{\ell}_i^{\text{role}} = W_r\mathbf{h}_i, \quad V_i = W_v\mathbf{h}_i, \tag{7}$$

where $\boldsymbol{\ell}_i^{\text{task}}$ and $\boldsymbol{\ell}_i^{\text{role}}$ parameterize categorical task/role distributions (with an always-valid idle slot), and V_i is the PPO value baseline. At inference, $\rho_i = \arg\max \boldsymbol{\ell}_i^{\text{role}}$. Infeasible or inactive tasks are masked in $\boldsymbol{\ell}_i^{\text{task}}$ by setting logits to $-\infty$ before normalization, and are scored by (8).

Feasibility- and load-aware decision making: At inference time we combine task logits with planner-derived ETAs via

$$s_{ij} = \eta_{ij}^{\text{norm}} - \alpha(\text{load})\,\lambda\,\tanh\big(\ell_{ij}/5\big), \tag{8}$$

where $\eta_{ij}^{\text{norm}} = ETA_{ij}/(\text{width} + \text{height})$, ℓ_{ij} denotes the task-head logit for assigning robot r_i to task t_j (derived from $\boldsymbol{\ell}_i^{\text{task}}$), and $\lambda = 0.4$. The global load indicator is

$$\text{load} = \frac{\sum_j d_j}{\max(|\mathscr{R}|, 1)}, \tag{9}$$

and the load factor is

$$\alpha(\text{load}) = \begin{cases} 1.0, & \text{load} \le 1.2, \\ \min\big(2.0,\ 1.0 + 0.4(\text{load} - 1.2)\big), & \text{otherwise}, \end{cases} \tag{10}$$

so network preference is emphasized as residual demand per robot increases, while $\tanh(\ell_{ij}/5)$ limits extreme logits. After score computation, a centralized non-differentiable assignment selects up to d_j robots per task by sorting s_{ij}, resolving conflicts by keeping each robot's best-scored task, and clipping to enforce capacity. This assignment is applied only during rollout/inference and is not part of the differentiable policy network.

PPO Training with Planning Feedback: We adopt a standard GAT encoder [14] and the clipped-surrogate PPO objective [15]. IAOM-specific components include (i) the intention-graph representation with residual demand and active commitments, (ii) feasibility masking with load-aware ETA scoring, and (iii) a centralized capacity-constrained assignment for scout–rescue MRTA.

During training, each robot samples task/role actions from the masked distributions induced by $\boldsymbol{\ell}_i^{\text{task}}$ and $\boldsymbol{\ell}_i^{\text{role}}$, while the centralized assignment enforces feasibility before environment execution. The value head V_i is used only as a baseline for advantage estimation and is not used in scoring or inference-time assignment.

Each robot receives a shaped reward

$$r_i = r_{\text{coop}} - \beta\,\eta_i^{\text{norm}} - \gamma\,w_i^{\text{wait}} + \kappa\log(1 + d_{t_i}) + r_{\text{pen}}, \tag{11}$$

with fixed coefficients $\beta = 2.0$, $\gamma = 0.3$, and $\kappa = 5.0$. Here $\eta_i^{\text{norm}} = \text{ETA}_i/(\text{width} + \text{height})$ uses the A$*$-path length when available (otherwise Manhattan distance); $w_i^{\text{wait}} = \min(\text{wait_time}/50,\ 3.0)$ penalizes prolonged waiting and capping the waiting penalty at 3.0 to avoid dominating cooperative rewards; r_{coop} is $+2.0$ for exact fulfillment of the residual demand at t_i, $+0.5$ for partial progress, and -2.0 for overfilling; and $r_{\text{pen}} = -5.0$ for assigning a robot to an invalid/inactive task, -1.0 for choosing *idle* when unmet demand exists, and 0 otherwise. This reward couples intention optimization to planner-derived travel cost, waiting, and residual-demand reduction under feasibility constraints.

D. Path Planning and Execution

After intention optimization, task–role assignments are grounded by a planning layer that converts high-level intentions into feasible motions. For each robot r_i, the final intention $\mathbf{z}_i^* = [r_i, t_i, \eta_i, \rho_i, \delta_i]$ is augmented with a path $\mathscr{P}_i$ and refined ETA $\tilde{\eta}_i$ on the grid graph $\mathbf{M} = (\mathscr{C}, \mathscr{E}_M)$:

$$\mathscr{P}_i = \text{A}^*(\mathbf{p}_i, \mathbf{p}_{t_i}), \quad \tilde{\eta}_i = |\mathscr{P}_i|, \tag{12}$$

and $\tilde{\eta}_i$ is written back to the corresponding node attribute, closing the loop between scheduling and motion feasibility. The framework is planner-agnostic (A$*$ can be replaced by cooperative A$*$ or conflict-based search [26]), but in our experiments we use a multi-agent A$*$ with task-priority ordering (robots expanded by increasing ETA to their assigned task) and a corridor-capacity constraint that forbids more than X (We set X=3) robots occupying a narrow corridor cell per time step. Optimized intentions and paths are broadcast for decentralized execution, where each robot follows

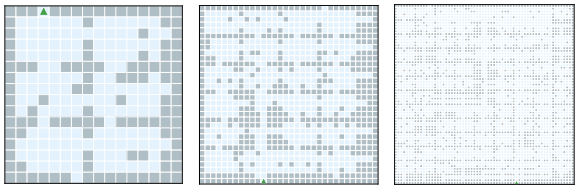
IV. Experiments

A. Experimental Setup

We reuse the grid-based environment and A$*$-based motion model from Section III-A; Fig. 3 illustrates representative 16×16, 32×32, and 64×64 maps, on which the number of tasks per episode typically ranges from about 7-20, 30-80, and 150-320, respectively.

Training setup: For each map size we train and test IAOM separately. The number of robots is set as a multiple of $|\mathscr{T}|$ to model three regimes: *abundant* ($|\mathscr{R}| = 2.0\,|\mathscr{T}|$), *normal* ($|\mathscr{R}| = 1.5\,|\mathscr{T}|$), and *tight* ($|\mathscr{R}| = 1.0\,|\mathscr{T}|$). Per size we generate $20/40/20$ (abundant/normal/tight) instances and use the 80-map union as a mixed training library. All trainable methods are run with three seeds $\{0, 1, 2\}$ and 5 outer training epochs per size.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.



(a) MAP 1 (b) MAP 2 (c) MAP 3

Fig. 3. Three procedurally generated 16×16, 32×32, and 64×64 search-and-rescue maps. Dark cells are obstacles, light cells are free, and the green triangle marks the entrance.

IAOM employs PPO with a replay buffer and windowed updates: in each outer epoch we collect trajectories into a buffer (capacity 30,000), run 6 PPO update epochs on this snapshot, then clear the buffer; baselines share the same outer-epoch budget with their own update rules. After training, each seed is validated on 10 maps per regime (30 maps total), and the best seed (lowest mean makespan) is kept. Final results are reported on disjoint test sets of 10 maps per regime and size (mean $\pm$ std).

Unless otherwise stated, IAOM uses a two-layer edge-aware GAT encoder (first layer: 4 heads, second: 1 head) with three linear heads for task logits (including an idle slot), role logits, and state value. Edge features have dimension 4 and store non-negative ETA, role id, task level, and a pending flag as raw scalars. PPO uses $\gamma = 0.99$, clipping range $\epsilon = 0.2$, value-loss coefficient 0.5, entropy coefficient 0.01, and batch size 192 samples per update epoch.

Baselines: We compare IAOM with five schedulers spanning greedy, optimization-based, heuristic, and learning-based paradigms. All baselines are adapted to the same grid-based, homogeneous rescue setting and share a unified interface: decisions are expressed as intentions $[r_i, t_i, \eta_i, \rho_i, \delta_i]$; only rescue intentions are optimized; tasks must be scouted and active with $L_j > 0$; ETA at evaluation uses A$*$; capacity is represented by residual demand; an idle option is always available. Learning-based baselines use the same training maps, density regimes, and overall training budget as IAOM.

- CHA (Hungarian) [6]: Expands task capacity into slots and solves a Hungarian matching on a robot-slot cost matrix using the same travel-cost model as IAOM, so capacity handling and cost evaluation are fully aligned.
- Greedy: In a fixed robot order, assigns each idle robot to the task with remaining demand that minimizes the same A$*$ travel cost; capacity and eligibility rules are identical to IAOM, differing only in the myopic selection rule.
- MSMRTA [7]: Streamlined team heuristic from Osooli et al. for the homogeneous case: forms local teams from nearby robots and scores them by travel cost plus a light cooperation term, with score components and capacity constraints matched to IAOM.
- HeteroRL [8]: Adapted heterogeneous MRTA policy with homogenized capabilities, outputting scores over robot-task-slot triples; at inference these scores are fused with IAOM-consistent travel costs and passed to a Hungarian matcher under capacity-slot expansion, using additional state flags to capture the reconnaissance-rescue task lifecycle.
- ScheduleNet [9]: Compact imitation-learning scheduler based on a robot-task GAT, adapted from ST-SR to our ST-MR setting; it is trained on expert labels from capacity-expanded Hungarian assignments with IAOM's travel costs, and at inference its scores are wrapped by the same capacity-aware pruning (top-k by residual need, conflict resolution, capacity clamping, idle backfill).

All experiments are run on a Linux workstation with an Intel Xeon-class CPU and an NVIDIA RTX 5090 GPU.

B. Evaluation Metrics and Results

We use three metrics to evaluate task responsiveness and resource efficiency:

- Mission Makespan (MM): total number of simulation steps until all tasks are completed.
- Average Completion Time (ACT): mean number of steps from task scouted to the moment its rescue requirement is fully satisfied.
- Robot Load Balance (RLB): standard-deviation-to-mean ratio of completed tasks per robot where each cooperative task is counted once for every participating robot; smaller values indicate a more balanced workload.

The best value for each metric is highlighted in **bold**. Results are reported as mean $\pm$ standard deviation over the evaluated test instances (with N specified for each table, typically 10 maps per regime).

Overall performance: Table I reports MM, ACT and RLB on three map sizes and three resource-density regimes (N=10 maps per regime). In the *abundant* and *normal* settings, IAOM consistently attains the lowest MM and ACT across all sizes. On 64×64 maps (abundant), for example, IAOM reduces MM from 281.2 (CHA) to 222.4 and ACT from 9.23 to 6.31; on 32×32 maps its ACT is roughly 40-50% lower than CHA in both abundant and normal regimes. Overall, across abundant/normal cases IAOM yields up to about 20% shorter makespan and up to about 50% lower completion time than the strongest optimization baseline, while keeping RLB comparable to the most balanced heuristics.

Although CHA employs the Hungarian algorithm, its optimality is limited to the current assignment round under known costs. Since task requirements are revealed progressively through reconnaissance, future coalition demand is unknown and mission-level makespan optimality is not guaranteed. Optimization- and heuristic-based baselines therefore react mainly to currently available information, while learning-based baselines do not explicitly model residual demand or active commitments. By encoding intentions, ETA information, and coalition requirements, IAOM better coordinates future task fulfillment and achieves lower MM and ACT.

In the *tight* regime, where robots roughly match task demand, no method dominates all metrics. HeteroRL achieves the smallest MM on all three sizes; ScheduleNet gives the best ACT at 16×16, while MSMRTA leads ACT on 32×32 and 64×64. RLB is likewise split (MSMRTA on 16×16, IAOM on 32×32, ScheduleNet on 64×64). IAOM remains competitive—its MM stays within a few steps of the best method on each size—but it no longer provides the uniformly best speed-balance trade-off

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.

TABLE I
OVERALL PERFORMANCE ON THREE MAP SIZES AND RESOURCE-DENSITY REGIMES

MAP SET	Method	Abundant			Normal			Tight		
		MM	ACT	RLB	MM	ACT	RLB	MM	ACT	RLB
16*16 N=10	CHA	51.3(±16.941)	3.144(±2.742)	0.927(±0.163)	56.2(±17.719)	6.732(±4.294)	0.656(±0.206)	74.1(±41.884)	16.041(±21.723)	0.365(±0.139)
	Greedy	51.5(±16.812)	3.187(±2.785)	0.979(±0.206)	55.8(±17.606)	7.189(±4.384)	0.700(±0.247)	74.1(±41.884)	15.851(±21.791)	0.365(±0.139)
	MSMRTA	51.3(±16.935)	3.144(±2.737)	**0.901(±0.163)**	57.9(±16.742)	7.300(±4.395)	**0.640(±0.198)**	65.5(±25.688)	8.073(±5.450)	**0.332(±0.123)**
	ScheduleNet	57.6(±16.728)	5.987(±3.625)	1.002(±0.299)	55.2(±16.461)	4.130(±2.424)	0.756(±0.272)	62.8(±27.331)	**7.089(±7.474)**	0.409(±0.234)
	HeteroRL	52.8(±14.811)	4.702(±2.716)	0.977(±0.272)	51.8(±14.169)	4.071(±2.332)	0.701(±0.274)	**59.6(±24.808)**	7.836(±8.982)	0.377(±0.230)
	IAOM	**49.8(±17.623)**	**2.445(±2.943)**	0.965(±0.218)	**49.9(±17.570)**	**3.227(±4.241)**	0.691(±0.220)	65.3(±30.220)	9.055(±10.479)	0.355(±0.172)
32*32 N=10	CHA	104.6(±13.537)	4.767(±2.006)	0.967(±0.115)	140.9(±31.536)	6.935(±2.261)	0.616(±0.177)	140.4(±18.839)	17.181(±8.086)	0.417(±0.049)
	Greedy	105.1(±14.453)	4.140(±1.709)	1.017(±0.116)	144.9(±29.012)	6.464(±3.100)	0.651(±0.194)	143.6(±11.439)	18.810(±7.590)	0.425(±0.049)
	MSMRTA	104.6(±13.537)	4.646(±1.884)	0.962(±0.111)	141.0(±31.417)	6.764(±2.297)	0.618(±0.179)	132.1(±15.326)	**9.621(±3.916)**	0.424(±0.055)
	ScheduleNet	111.2(±14.675)	9.195(±4.791)	**0.903(±0.083)**	143.7(±34.825)	7.399(±2.093)	**0.546(±0.205)**	131.3(±15.284)	12.065(±3.397)	0.422(±0.057)
	HeteroRL	104.6(±13.537)	4.762(±1.932)	0.965(±0.110)	141.0(±31.417)	6.705(±2.254)	0.615(±0.175)	**129.2(±18.384)**	14.638(±5.236)	0.418(±0.054)
	IAOM	**93.4(±11.595)**	**2.024(±1.070)**	0.966(±0.097)	**128.1(±20.849)**	**3.873(±3.502)**	0.688(±0.153)	134.4(±20.824)	15.403(±6.095)	**0.408(±0.043)**
64*64 N=10	CHA	281.2(±35.555)	9.226(±1.620)	0.957(±0.067)	279.6(±38.604)	11.018(±2.866)	0.636(±0.076)	354.2(±26.653)	39.398(±8.427)	0.449(±0.027)
	Greedy	309.1(±42.646)	9.062(±1.607)	1.030(±0.072)	311.8(±34.008)	11.120(±1.872)	0.712(±0.070)	353.3(±26.647)	39.427(±8.428)	0.452(±0.027)
	MSMRTA	279.5(±35.169)	9.300(±1.617)	0.959(±0.057)	283.4(±40.384)	11.316(±2.908)	0.631(±0.072)	328.5(±29.261)	**19.493(±2.793)**	0.434(±0.028)
	ScheduleNet	295.1(±45.689)	19.492(±9.744)	**0.851(±0.062)**	276.4(±31.604)	12.330(±2.563)	**0.528(±0.109)**	328.2(±22.907)	25.585(±3.794)	**0.408(±0.034)**
	HeteroRL	278.8(±35.487)	9.124(±1.629)	0.960(±0.054)	283.4(±40.384)	11.129(±2.881)	0.630(±0.073)	**327.9(±25.330)**	30.894(±4.880)	0.413(±0.033)
	IAOM	**222.4(±22.119)**	**6.313(±1.964)**	0.936(±0.092)	**230.9(±47.780)**	**10.209(±5.866)**	0.630(±0.108)	330.8(±35.791)	36.372(±6.040)	0.428(±0.031)

Note: Bold, shaded entries indicate the best result for each metric.

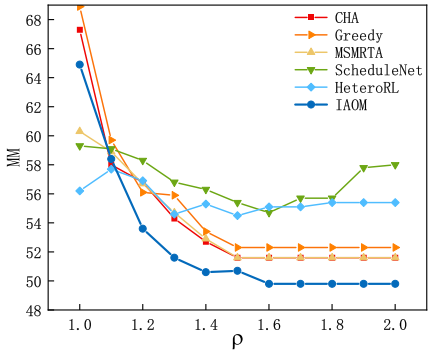


Fig. 4. Mission makespan (MM) on 16×16 test maps (N=10) as the robot-task ratio $\rho = |\mathscr{R}|/|\mathscr{T}|$ increases.

TABLE II
AVERAGE PER-ROUND DECISION TIME (S) ON 32×32 (NORMAL REGIME, N=10)

Method	Decision time [s]
CHA	0.0578(±0.0645)
Greedy	0.0530(±0.0611)
MSMRTA	0.0542(±0.0622)
ScheduleNet	**0.0464(±0.0432)**
HeteroRL	0.0550(±0.0586)
IAOM	0.0542(±0.0561)

TABLE III
ABLATION OF PLANNER-DERIVED ETA AND LOAD-AWARE SCORING ON 32×32 MAPS (NORMAL REGIME, N=10)

Method	MM	ACT	RLB
IAOM	**128.1(±20.849)**	**3.873(±3.502)**	0.688(±0.153)
IAOM (no ETA)	131.3(±24.495)	4.392(±4.189)	0.694(±0.154)
IAOM (no load)	144.4(±28.472)	8.526(±1.594)	**0.572**(±0.152)

under extreme scarcity. This behaviour is consistent with the structure of the problem: when almost every robot is already committed, there is often at most one feasible team per task, so coordinating ETAs and coalition sizes offers limited additional leverage and simple slot-based matchers that directly fill residual capacity approach the constrained optimum. In addition, our reward shaping places relatively stronger emphasis on reducing ACT and prioritizing high-demand tasks, which can tolerate a small increase in MM or RLB in tight regimes in exchange for shorter per-task delays.

Fig. 4 further varies the robot–task ratio $\rho = |\mathscr{R}|/|\mathscr{T}|$ from 1.0 to 2.0 on 16×16 maps. All methods benefit from increasing ρ, but IAOM's MM drops fastest and stabilizes around 50 steps once $\rho \gtrsim 1.5$, while CHA, Greedy, and MSMRTA plateau several steps higher and ScheduleNet remains clearly above them. At $\rho \approx 1.0$ all methods are constrained by hard resource limits; once modest slack appears, IAOM can exploit the intention graph to coordinate coalitions and avoid redundant motion, turning extra robots into effective makespan reduction rather than additional congestion.

Across settings, PPO training converges stably, with moderately slower convergence on larger maps. Training time increases sharply with map size—remaining below one second up to 50×50, rising to several seconds at 60×60, and exceeding ten seconds at 80×80. Inference per allocation round remains negligible, requiring only a single forward pass.

Decision-time comparison: To assess computational overhead, Table II reports the average per-round decision time on 32×32 maps in the normal regime. All methods take about 50–60 ms per coordination round, with IAOM (0.054 s) comparable to CHA and Greedy and only slightly slower than ScheduleNet (0.046 s). This indicates that the intention graph and GAT-PPO policy do not introduce prohibitive runtime compared with optimization- or heuristic-based schedulers.

Ablation on execution coupling: Table III ablates two key design choices on 32×32 maps (normal). Removing planner-derived ETA from the graph and reward ("no ETA") slightly degrades MM and ACT, whereas disabling load-aware scoring ("no load") leads to a much larger increase in ACT (from 3.87 to 8.53 steps on average) and a noticeable MM increase. These trends confirm that both ETA information and load-aware decoding contribute to IAOM's execution-aware behavior, with load-aware scoring having the strongest impact on responsiveness.

Sensitivity to initial intentions: IAOM is trained with randomly generated rescue intentions that are then refined by the GAT–PPO policy. To assess sensitivity to this initializer, we freeze the policy and regenerate initial intentions at test time using Greedy, Random (default), and CHA-based matching (Table IV). Across all regimes on 16×16 maps, the three variants yield almost identical MM/ACT and only minor RLB differences, suggesting that IAOM's performance is driven mainly by intention-graph refinement rather than a carefully engineered initialization rule.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
SENSITIVITY OF IAOM TO DIFFERENT INITIAL-INTENTION HEURISTICS ON 16×16 MAPS (N=10 PER REGIME)

	MM			ACT			RLB		
Regime	Greedy	Random	CHA	Greedy	Random	CHA	Greedy	Random	CHA
abundant	53.0	**49.8**	51.3	3.38	**2.44**	3.32	0.98	0.96	**0.91**
normal	**49.6**	49.9	**49.6**	**3.12**	3.23	3.34	0.70	0.69	**0.64**
tight	65.4	65.4	65.4	9.57	9.57	9.57	0.36	0.36	**0.32**

TABLE V
RECONNAISSANCE-AWARE, NO-RETURN ABLATION ON 16×16 MAPS (N=10 PER REGIME)

	Abundant		Normal		Tight	
Method	MM	ACT	MM	ACT	MM	ACT
CHA	**25.4**	1.70	**25.4**	1.70	31.5	5.84
Greedy	25.6	1.99	25.6	1.99	33.0	6.31
MSMRTA	26.0	1.81	26.0	1.81	25.7	2.49
ScheduleNet	36.3	5.86	32.3	4.27	31.5	3.52
HeteroRL	26.0	1.81	26.0	1.81	26.9	3.08
IAOM	26.0	**1.41**	26.0	**1.41**	**25.3**	**2.45**

Effect of the return stage: Table V evaluates IAOM when the return phase is disabled while keeping reconnaissance-driven task revelation. In this pure rescue-allocation setting, all methods obtain very similar makespans (about 25-26 steps) under abundant/normal regimes and the overall gap narrows, but IAOM still achieves the lowest ACT in all regimes and the best MM in the tight case. This indicates that IAOM's advantage does not stem from the modeling of the return stage alone.

V. DISCUSSION AND CONCLUSION

This letter introduced IAOM, an intention-aware optimization model for dynamic multi-robot cooperative task allocation. By representing robot intentions (target, role, ETA) as execution-level coordination variables and organizing them into an intention graph, IAOM couples high-level allocation with motion feasibility through an edge-aware GAT-PPO policy and an A$*$-based execution layer. Across multiple map sizes and resource regimes, IAOM improves makespan and completion time under abundant and normal conditions while remaining competitive in tight regimes.

Beyond performance gains, IAOM highlights intention as a coordination abstraction that bridges abstract MRTA decisions and execution in cluttered environments. Although evaluated with a centralized scheduler, the intention representation itself is topology-agnostic and can extend to hierarchical or partially distributed architectures.

In real deployments, coordination quality will depend on ETA accuracy and communication latency, and centralized schemes may face computational and bandwidth bottlenecks for large teams or scarce resources. These constraints affect the coordination topology rather than the intention abstraction. Since IAOM relies only on task/robot attributes and planner-derived travel-time estimates, it is not restricted to grid maps and can operate with alternative environment models (e.g., roadmaps or navigation meshes). Extending IAOM toward scalable decentralized implementations and more realistic execution models remains important future work.

REFERENCES

[1] K. Nagatani et al., "Multi-robot exploration for search and rescue missions: A report of map building in robocuprescue 2009," in *Proc. IEEE Int. Workshop Saf., Secur. Rescue Robot.* 2009, pp. 1–6.

[2] R. R. Murphy, *Disaster Robot.*, Cambridge, MA, USA: MIT Press, 2014.

[3] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 25, no. 3, pp. 225–241, 2006.

[4] M. G. Lagoudakis et al., "Auction-based multi-robot routing," in *Proc. Robot.: Sci. Syst.*, Rome, Italy, 2005, vol. 5, pp. 343–350.

[5] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, no. 2, pp. 220–240, Apr. 1998.

[6] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, no. 1–2, pp. 83–97, 1955.

[7] H. Osooli, P. Robinette, K. Jerath, and S. R. Ahmadzadeh, "A multi-robot task assignment framework for search and rescue with heterogeneous teams," 2023, *arXiv:2309.12589*.

[8] W. Dai, U. Rai, J. Chiun, Y. Cao, and G. Sartoretti, "Heterogeneous multi-robot task allocation and scheduling via reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 10, no. 3, pp. 2654–2661, Mar. 2025.

[9] Z. Wang, C. Liu, and M. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints," *Auton. Robots*, vol. 46, no. 1, pp. 249–268, 2022.

[10] S. Paul, P. Ghassemi, and S. Chowdhury, "Learning scalable policies over graphs for multi-robot task allocation using capsule attention networks," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 8815–8822.

[11] W. Dai, A. Bidwai, and G. Sartoretti, "Dynamic coalition formation and routing for multirobot task allocation via reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 16567–16573.

[12] Z. Lu et al., "A reinforcement learning-based optimization method for task allocation of agricultural multi-robots clusters," *Comput. Elect. Eng.*, vol. 120, 2024, Art. no. 109752.

[13] Z. Wu, "A multi-robot task allocation method based on graph attention network and unsupervised learning," in *Proc. IEEE Int. Conf. Unmanned Syst.*, 2024, pp. 1222–1227.

[14] P. Veličković et al., "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, vol. 6, no. 2, 2018.

[15] J. Schulman et al., "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[16] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[17] S. Choudhury et al., "Dynamic multi-robot task allocation under uncertainty and temporal constraints," *Auton. Robots*, vol. 46, no. 1, pp. 231–247, 2022.

[18] Y. Emam et al., "Adaptive task allocation for heterogeneous multi-robot teams with evolving and unknown robot capabilities," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 7719–7725.

[19] V. M. Tuck et al., "SMT-based dynamic multi-robot task allocation," in *Proc. NASA Formal Methods Symp.*, 2024, pp. 331–351.

[20] G. Neville, S. Chernova, and H. Ravichandar, "D-ITAGS: A dynamic interleaved approach to resilient task allocation, scheduling, and motion planning," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 1037–1044, Feb. 2023.

[21] C. R. Garrett et al., "Integrated task and motion planning," *Annu. Rev. Control Robot., Auton. Syst.*, vol. 4, no. 1, pp. 265–293, 2021.

[22] J. S. Park, C. Park, and D. Manocha, "I-Planner: Intention-aware motion planning using learning-based human motion prediction," *Int. J. Robot. Res.*, vol. 38, no. 1, pp. 23–39, 2019.

[23] S. Qi and S.-C. Zhu, "Intent-aware multi-agent reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7533–7540.

[24] W. Kim, J. Park, and Y. Sung, "Communication in multi-agent reinforcement learning: Intention sharing," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 12440–12454.

[25] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, "Spatial intention maps for multi-agent mobile manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8749–8756.

[26] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on September 09,2026 at 16:11:25 UTC from IEEE Xplore. Restrictions apply.