IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Learning Policies for Dynamic Coalition Formation in Multi-Robot Task Allocation

Lucas C. D. Bezerra[1], Ataíde M. G. dos Santos[2], and Shinkyu Park[1]

***Abstract*—We propose a decentralized, learning-based framework for dynamic coalition formation in Multi-Robot Task Allocation (MRTA). Our approach extends MAPPO by integrating spatial action maps, robot motion planning, intention sharing, and task allocation revision to enable effective and adaptive coalition formation. Extensive simulation studies confirm the effectiveness of our model, enabling each robot to rely solely on local information to learn timely revisions of task selections and form coalitions with other robots to complete collaborative tasks. The results also highlight the proposed framework's ability to handle large robot populations and adapt to scenarios with diverse task sets.**

***Index Terms*—Multi-Robot Systems, Cooperating Robots, Reinforcement Learning.**

I. Introduction

COALITION formation, widely observed in nature, such as in ant colonies and beehives, inspires researchers and engineers to design simple robots capable of strategic cooperation. In multi-robot systems, coalitions emerge when the collective actions of robots exceed what they could accomplish individually, as illustrated in Fig. 1. In this work, we focus on developing policies for a team of robots capable of performing tasks that require coalition in dynamic environments. In such scenarios, robots must learn to coordinate by leveraging local information gathered from their own sensors and shared data from neighboring robots to effectively execute tasks.

Multi-Robot Task Allocation (MRTA) is a research theme directly relevant to our work, focusing on the challenge of assigning and executing multiple tasks across a team of robots to optimize a collective objective over an extended time horizon. We focus on MRTA problems where: (i) each robot is capable of executing only one task at a time, (ii) tasks may require multiple robots to collaborate simultaneously for completion, (iii) robots must be adjacent to a task to execute it, necessitating movement to the task location beforehand, and (iv) completed tasks are removed, and new tasks continuously spawn in the environment. This problem formulation falls under the Single-Task robots, Multi-Robot tasks, Time-extended Assignment (ST-MR-TA) category of MRTA, as defined by the taxonomy in [1]. Relevant applications include transportation [2] and service dispatch scenarios [3], [4] such as firefighting and large-scale disaster response.

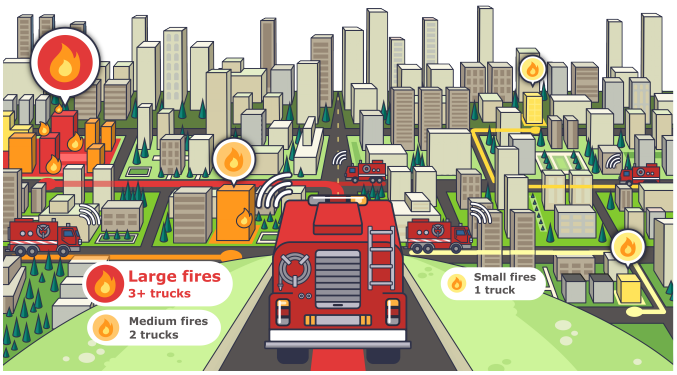


Fig. 1: Firetruck dispatch as an illustrative example of a dynamic coalition formation problem: a team of firetrucks must form coalitions to extinguish fires that cannot be accomplished individually. In the event of a catastrophe that disrupts communication infrastructure, firetrucks may need to independently determine which fires to prioritize.

We focus on developing a decentralized solution where each robot observes only its immediate surroundings. With such partial observability in MRTA, robots only identify tasks within their sensor range, causing the set of perceived tasks to change as they move. Additionally, we consider dynamic environments where new tasks continuously emerge over time. Well-established approaches to partially observable task allocation problems rely on inter-robot communication, where robots share their observations, internal state, and decision with their neighbors. Market-based approaches [5], [6] are among the most popular frameworks due to their robustness and scalability. However, these methods often rely on high-frequency communication between robots to reach agreement before task assignments are made [6]–[8]. Although this strategy is efficient in some restricted cases, it becomes impractical in dynamic environments.

We propose a learning-based framework for multi-robot dynamic coalition formation to address problems within the ST-MR-TA MRTA category. In this framework, each robot utilizes a receding horizon-type planning strategy to generate a long-term plan for task execution, which is repeatedly revised as necessary in subsequent time steps. Each robot shares its plan with other robots that are within its communication range. This combination of task reassessment and sharing of the task execution plan enables dynamic coalition formation. Our contributions are summarized as follows:

(i) To the best of our knowledge and based on a recent survey article [9], the problem of decentralized dynamic coali-

Manuscript received: Feburary, 22, 2025; Revised May, 11, 2025; Accepted June, 30, 2025.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by funding from King Abdullah University of Science and Technology (KAUST).

[1] Bezerra and Park are with Division of Computer, Electrical and Mathematical Sciences and Engineering (CEMSE), King Abdullah University of Science and Technology (KAUST), Thuwal, 23955-6900, Kingdom of Saudi Arabia. {lucas.camaradantasbezerra, shinkyu.park}@kaust.edu.sa

[2] Santos is with Department of Electrical Engineering, Federal University of Sergipe (UFS), São Cristóvão, Sergipe, 49107-230, Brazil. ataidegualberto.eng@gmail.com

tion formation under partial observability has not been previously addressed. To facilitate coalition, we extend a Multi-Agent Reinforcement Learning (MARL) algorithm by integrating spatial action maps, robot motion planning, intention sharing, and task allocation revision. Compared to existing MARL approaches, such as [10]–[12], the integration of these key components fosters coalition formation through decentralized decision-making in dynamic environments (see Table IIc for a comprehensive comparison).

(ii) Each robot's policy is modeled as an end-to-end convolutional neural network based on the U-Net architecture [13]. The network processes information on nearby tasks and other robots within the robot's sensing range and integrates the intentions of neighboring robots. It is trained using the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm with the parameter-sharing technique.

(iii) Through extensive simulations, we demonstrate that our approach outperforms existing methods including a market-based approach, Project manager-oriented Coalition Formation Algorithm (PCFA). We also demonstrate its scalability with up to 1000 robots and its generalizability across a wide range of task settings.

This paper is organized as follows: Section II reviews related work in MRTA and MARL. In Section III, we formally define the problem of dynamic coalition formation as a Dec-POMDP. Section IV outlines our proposed method as a solution to the problem. In Section V, we evaluate the effectiveness, scalability, and generalizability of the proposed method through extensive simulations, comparing it to the baseline. Section VI summarizes the paper and outlines future research directions.

II. Related Work

MRTA is widely researched topic at the intersection of robotics and multi-agent systems [6]–[8], [14]–[23]. Among the various problem settings, ST-MR-TA MRTA formulations are most relevant to our work. Solutions to ST-MR-TA MRTA include optimization-based [16]–[19], trait-based [21], [22], and market-based [6]–[8], [23] methods. Among decentralized solutions for MRTA, market-based approaches are particularly popular. Specifically, PCFA [8] is a market-based solution designed for coalition formation in homogeneous teams with dynamic tasks. Due to the similarity in problem formulation, we selected PCFA as the baseline for our simulations reported in Section V.

Reinforcement Learning (RL) focuses on learning a policy (i.e., a mapping from observation to action) that maximizes a robot's total reward in a Markov decision process. It has been widely applied to various robotics problems, particularly in object manipulation [11], [24]–[26]. While many approaches aim to directly control robot actuators, studies such as [11], [24], [27] propose the use of an abstract action map. In this framework, the policy selects a task at each time step, while a motion planner handles the low-level actuator control to navigate the robot to the task location. This abstraction alleviates the learned policy from handling low-level control, allowing it to concentrate on long-term planning.

This approach is extended to the multi-robot domain in [12], where robots not only select task locations but also communicate their chosen task locations to nearby robots. The authors show that this additional communication improves cooperation and enhances overall performance. However, their method does not address coalition formation. As demonstrated in Section V, their approach, with appropriate adaptations, performs suboptimally compared to ours in scenarios that require effective coalition formation. MARL naturally extends the conventional RL framework to cooperative multi-robot scenarios, where instead of learning policies to maximize individual rewards, robots learn to maximize a global reward through coordination [28]. Several studies have adapted RL algorithms for the multi-robot domain [29]–[32]. A widely adopted approach is the Centralized Training and Decentralized Execution (CTDE) paradigm, where robots are trained using shared information [30]–[32]. CTDE offers faster training and addresses non-stationarity.

Several studies have applied MARL to MRTA [10], [12], [31], [33]. For example, [31], [33] specifically address MRTA problems involving multi-robot tasks. The MRTA problem was introduced to the MARL community as a benchmark in [31], where the authors released a Python package called Level-Based Foraging (LBF). In LBF, each object is assigned a level, and it can only be picked up by a group of robots if the sum of the robots' levels meets or exceeds the object's level. Although LBF has become a popular MARL benchmark [34]–[36], it lacks support for dynamic object spawning and abstract action spaces (e.g., selecting task positions on the grid). In [34], several MARL algorithms were evaluated across popular benchmarks, including LBF (in fully observable scenarios only), and the EPyMARL package was released for community use. We expand their implementation by incorporating robot communication and additional features, which are detailed further in Section IV.

In [10], [33], MARL is applied to an extended version of MRTA, where robots must not only collect objects but also transport them. In [33], which involves multi-robot tasks, the authors propose a novel inter-robot communication architecture. However, their approach assumes a fully observable environment. In [10], the environment is defined using a multi-channel, grid-like observation space under partial observability, with the robots' policy modeled using a convolutional neural network, similar to our approach. However, unlike our work, they do not employ an abstract action space and use a shallower policy architecture.

III. Problem Description

Consider a rectangular 2D grid environment of size $W \times W$, bounded by walls on all sides, where robots and tasks are distributed, as illustrated in Fig. 2. Each cell p in the grid belongs to the cell space $\mathbb{P} = \{1, \dots, W\} \times \{1, \dots, W\}$ and can be either empty or occupied by a robot, a task, or a wall. The grid contains a homogeneous team of N robots, where the position of the i-th robot at time step t is denoted as $p_{i,t} \in \mathbb{P}$. At each t, a robot can either remain in its current position or move to an adjacent cell, including diagonal moves. Robots

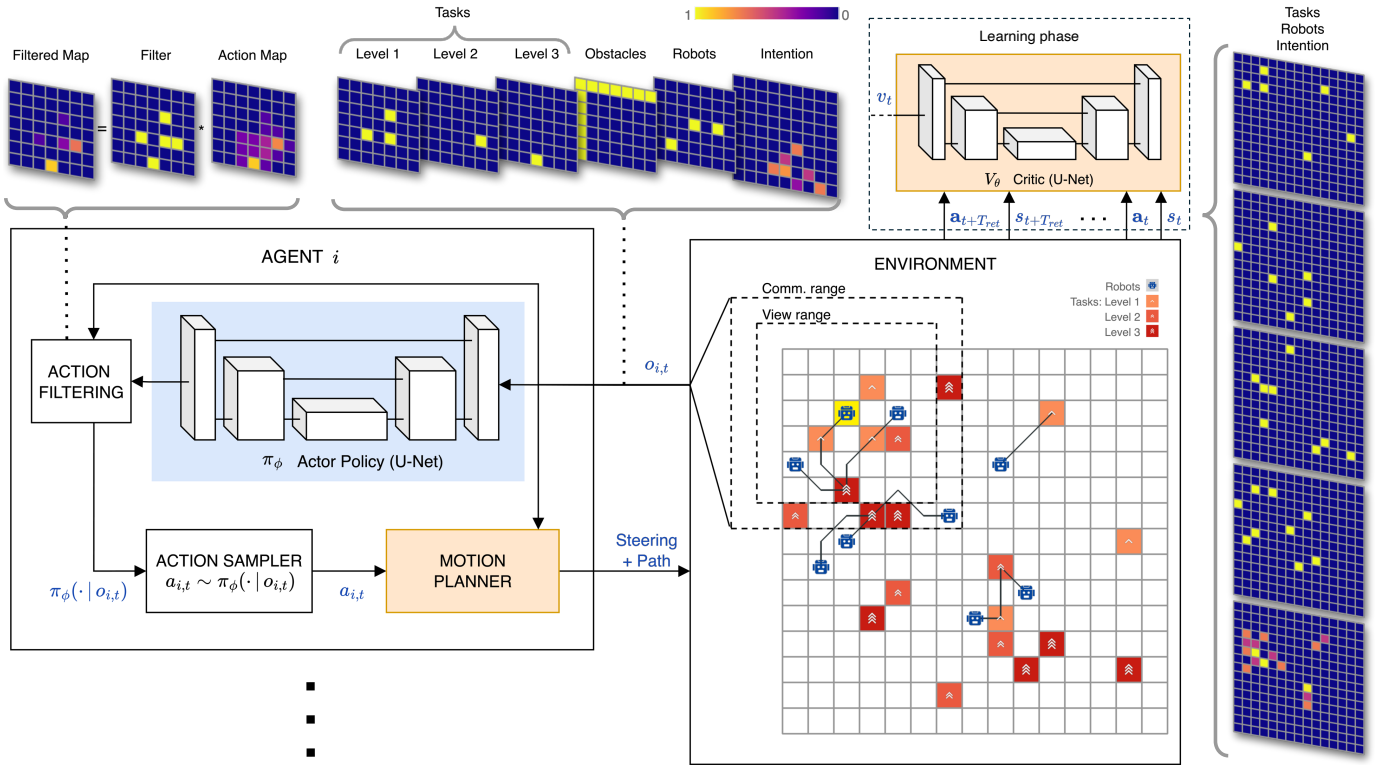


Fig. 2: MRTA Environment-Robot Interaction: On the right, a depiction of the environment is shown along with the state representation s_t used by the critic during training. Each robot receives an observation $o_{i,t}$ consisting of multiple channels. The observation is then fed into the policy π_ϕ and filtered to consider only visible tasks. A task is selected from the policy output, and the motion planner finds a feasible path to it.

observe the cells within their view range, defined by R_{view} cells, and communicate with others within a communication range of R_{comm} cells.

Each task is associated with a positive integer $l_i \in \{1, \ldots, L_{max}\}$, referred to as the task's *level*, which represents the number of robots required to execute it. Here, L_{max} denotes the maximum task level. We consider a dynamic task set: tasks are removed from the environment once completed by robots, and new tasks can appear at any time step in unoccupied cells. At time step t, the set of available tasks is represented as $\mathbb{T}_t = \{(q_i, l_i)\}_{i=1}^{T_t}$, where q_i denotes the location of the i-th task, l_i indicates its level, and T_t represents the total number of tasks available.

We consider two task spawn settings: (i) *Bernoulli (p) spawn*: At each time step t, a new task can appear in any unoccupied cell with probability p. The level of each newly spawned task is sampled from a discrete uniform distribution $\mathcal{U}(\{1, \ldots, L_{max}\})$. (ii) *Instant respawn*: When a task is completed, a new task of the same level immediately respawns in one of the unoccupied cells. This setting maintains a constant task density throughout the episode.

To design a decentralized policy under partial observability, we model the problem as a Decentralized Partially-Observable Markov Decision Process (Dec-POMDP) [37], [38]. A Dec-POMDP is defined by the tuple $(N, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, O, R)$, where N is the number of robots, $\mathcal{S}$ represents the state space of the environment, $\mathcal{A}$ and $\mathcal{O}$ denote the action and observation spaces of individual robots, respectively. The state transition function is defined as $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A}^N \to \mathbb{R}_+$, and the reward function as $R : \mathcal{S} \times \mathcal{A}^N \to \mathbb{R}$.[1]

Each robot i receives an observation $o_{i,t} \in \mathcal{O}$ at time t and independently selects an action $a_{i,t} \in \mathcal{A}$ based on its (stochastic) policy $\pi : \mathcal{A} \times \mathcal{O} \to \mathbb{R}_+$, i.e., $a_{i,t} \sim \pi(\cdot|o_{i,t})$. An observation $o_{i,t}$ is a function of the state $s_t \in \mathcal{S}$ and is defined as $o_{i,t} = O(s_t, i)$, where $O : \mathcal{S} \times \{1, \ldots, N\} \to \mathcal{O}$ is the observation function. Given the state s_t and the joint action $\mathbf{a}_t = (a_{1,t}, \ldots, a_{N,t}) \in \mathcal{A}^N$, the next state $s_{t+1} \in \mathcal{S}$ is sampled according to the state transition function $\mathcal{T}$, i.e., $s_{t+1} \sim \mathcal{T}(\cdot|s_t, \mathbf{a}_t)$. The transition function $\mathcal{T}$ captures both deterministic dynamics (e.g., robot movement, task execution) and stochastic dynamics (e.g., task spawning) of the environment. For each state-action pair, a shared reward $r_t = R(s_t, \mathbf{a}_t)$ is computed for all robots.

In a Dec-POMDP, state-action trajectories are represented as an episode $\xi := (s_0, \mathbf{a}_0, s_1, \mathbf{a}_1, \ldots, s_T)$. Given an initial state distribution $P_0 : \mathcal{S} \to \mathbb{R}_+$ and the episode likelihood $P_\pi(\xi) = P_0(s_0) \prod_{t=0}^{T-1} \prod_{i=0}^{N-1} \pi(a_{i,t}|o_{i,t}) \mathcal{T}(s_{t+1}|s_t, \mathbf{a}_t)$, the performance of a policy π can be evaluated using the expected episodic reward $\mathbb{E}_{\xi \sim P_\pi(\cdot)}\left[\sum_{t=0}^{T} R(s_t, \mathbf{a}_t)\right]$, where T is the terminal time. When a sufficiently large number of episodes is available in a dataset $\mathcal{D}$, this expectation can be approximated as $\mathbb{E}_{\xi \sim \mathcal{D}}[\sum_{t=0}^{T} R(s_t, \mathbf{a}_t)]$, thereby removing the need for explicit knowledge of P_0 and $\mathcal{T}$.

[1]Here, we provide a general description of a Dec-POMDP; implementation details specific to our framework are presented in Section IV-A.

TABLE I: Description of CNN input channels.

Notation	Space	Description
$\mathcal{C}_t^{\text{robot}}$	$\{0,1\}^{W\times W}$	Robots
$\mathcal{C}_t^{\text{task}}$	$\{0,1\}^{W\times W\times L_{max}}$	Tasks (separated by level)
$\mathcal{C}_t^{\text{obs}}$	$\{0,1\}^{W\times W}$	Obstacles
$\mathcal{C}_{i,t}^{\text{intent}}$	$\mathbb{R}_+^{W\times W}$	Robot i's intention map

IV. Task Allocation Policy for Dynamic Coalition Formation

We elaborate on the two main contributions of this work: (i) the design of decentralized task allocation policy structure, and (ii) the development of a learning-based algorithm within the Dec-POMDP framework to identify the optimal policy. Our proposed policies are designed to select tasks for execution, assess whether a robot's current task allocation requires revision, and determine new task assignments based on the robot's evolving local information. Once a task is assigned, each robot performs low-level control to move to a neighboring cell along a collision-free path within a single time step. In the subsequent time step, the policy reassesses the task allocation based on updated local information, redirecting the robot to a new task if necessary. This iterative process ensures that the overall collective objective is optimized.

A. Dec-POMDP and Task Allocation Policy

1) Reward Function: Higher-level tasks, which require coalitions of multiple robots, are more challenging to complete. To prevent robots from favoring lower-level tasks, we prioritize higher-level tasks by rewarding robots with a value equal to the square of the task level. Specifically, the reward at each time step t is defined as $r_t = \sum_{(q_i,l_i)\in\mathbb{T}_t\setminus\mathbb{T}_{t+1}} l_i^2$, where $\mathbb{T}_t\setminus\mathbb{T}_{t+1}$ represents the set of tasks completed at time step t, and l_i denotes the level of task i.[2]

2) State and Observation Spaces: We adopt a 2D occupancy grid representation segmented into multiple channels (see Table I), enabling the use of Convolutional Neural Network (CNN) architectures for policy design, similar to approaches in [10]–[12]. This differs from the representation used in LBF [31], which encodes the state as a vector of robots and their positions. In our framework, the state is defined as multiple separate channels as $s_t \triangleq (\mathcal{C}_t^{\text{robot}}, \mathcal{C}_t^{\text{task}}, \mathcal{C}_t^{\text{obs}}, \mathcal{C}_t^{\text{intent}})$, where $\mathcal{C}_t^{\text{intent}} = \mathcal{C}_{1,t}^{\text{intent}} \times \cdots \times \mathcal{C}_{N,t}^{\text{intent}}$.[3]

The observation $O(s_t, i)$ for robot i is derived by projecting and cropping s_t onto areas centered around the robot's position, determined by its view range R_{view} and communication range R_{comm}. Specifically, the robot intention channel $\mathcal{C}_t^{\text{intent}}$ includes $\mathcal{C}_{j,t}^{\text{intent}}$ from robot j if j is within R_{comm} cells of robot i. The other channels, $\mathcal{C}_t^{\text{robot}}, \mathcal{C}_t^{\text{task}}, \mathcal{C}_t^{\text{obs}}$, are obtained by cropping s_t according to R_{view}.

3) Action Space, Motion Planner, and Intention Map: Following a similar representation as in [11], [24], a robot's action – determined by the task allocation policy π_ϕ – is defined as the location of its next task. Robots can select task locations only if they are within its view range; if no tasks are within view range, all grid locations within communication range are available for action selection. This constraint provides prior information, reducing training time while encouraging robots to commit to specific tasks.

Algorithm 1: Robot Motion Control

Function `MotionPlanning`$(p_{i,t}, a_{i,t})$**:**
  Using the A* algorithm, find a collision-free path $(\bar{p}_1, \cdots, \bar{p}_n)$ where $\bar{p}_1 = p_{i,t}$ and $\bar{p}_n = a_{i,t}$.
  return $(\bar{p}_1, \cdots, \bar{p}_n)$

Procedure `NextRobotPosition`$(p_{i,t}, o_{i,t})$**:**
  $a_{i,t} \sim \pi_\phi(\cdot|o_{i,t})$ `// draw a task location using the learned policy` π_ϕ `with local observation` $o_{i,t}$.
  $(\bar{p}_1, \cdots, \bar{p}_n)$ = `MotionPlanning` $(p_{i,t}, a_{i,t})$
  Generate an intention map with $(\bar{p}_1, \cdots, \bar{p}_n)$ and share it with neighboring robots.
  $p_{i,t+1} \leftarrow \bar{p}_2$ `// assign` $\bar{p}_2$ `as the robot's next position to move to.`
  return $p_{i,t+1}$

After selecting a task location, a motion planner computes the shortest collision-free path to that location while avoiding other robots, obstacles, and other tasks. We adopt the A* search algorithm for simplicity. The path is dynamically recomputed whenever a potential collision is detected. Robots follow these updated paths to their assigned task locations. To improve coalition formation for higher-level tasks, the policy π_ϕ assigns actions by considering the long-term task execution plans of neighboring robots.

We also employ an *intention map* [12] that encodes each robot's task location and collision-free path. Inspired by the encoding scheme in [12], we use an exponentially decaying path representation, where the i-th position on the path is weighted by α^{-i}, with $\alpha = 2/3$ in our simulation, and embedded in the intention map. This intention map is shared among neighboring robots to facilitate coalition formation. Algorithm 1 outlines the motion control process for each robot guided by the learned policy π_ϕ.

B. Optimal Task Allocation Policy

1) MAPPO: To compute the optimal policy, we adopt MAPPO [30], a CTDE algorithm designed for MARL. MAPPO enables robots to share experiences during the learning phase, which is particularly beneficial for training policies in teams of homogeneous robots. MAPPO is an Actor-Critic algorithm comprising two distinct networks: the actor π_ϕ and the critic V_θ. The actor outputs a distribution over actions based on each robot's observations, formally defined by the policy $\pi_\phi : \mathcal{A} \times \mathcal{O} \to \mathbb{R}_+$, where ϕ represents the parameters of the policy network. These parameters ϕ are shared across all robots, a technique known as *parameter sharing* [39], which improves training efficiency, especially in cooperative scenarios involving homogeneous robots. The critic computes an approximate value function $V_\theta : \mathcal{S} \to \mathbb{R}$, parameterized by θ, to estimate the expected discounted sum of future rewards $\mathbb{E}[\sum_{\tau=t}^{T} \gamma^{\tau-t} R(s_\tau, \mathbf{a}_\tau)]$.

[2]Once the coalition requirement is met, we assume that the robots can complete each task within a single time step.

[3]To simplify the design and improve the efficiency of critic model training, our implementation consolidates $\mathcal{C}_t^{\text{intent}}$ into a single aggregated channel that combines the intention channels of all robots. Empirical studies indicate that this aggregation does not impact the training outcomes.

The critic V_θ is trained to accurately estimate future rewards, while the actor π_ϕ is trained to select actions that maximize the critic's estimate. In our framework, both the actor π_ϕ and the critic V_θ are implemented using U-Nets [13], a CNN-based architecture. For this implementation, the actor's input (observations) and output (action distribution) as well as the critic's input (state), are represented as images (3D arrays). The U-Net architecture is particularly advantageous due to its ability to capture both local spatial relationships and global context within the input image, making it well-suited for policy learning.

2) Learning Task Allocation and Revision: As shown in Fig. 2, we design a task allocation policy that assigns a task location $a_{i,t}$ to each robot based on its local information: $a_{i,t} \sim \pi_\phi(\cdot|o_{i,t})$, where $o_{i,t} = O(s_t, i)$. In a partially observable environment with dynamically spawning tasks, a robot may encounter new tasks and other robots as it moves, which were previously unaccounted for. This dynamic nature makes it challenging for multiple robots to form coalitions for higher-level tasks. Consequently, the policy must allow robots to revise their task execution plans when necessary. However, frequent task revisions can disrupt coalition formation and hinder the completion of higher-level tasks.

To address this challenge, we extend the MAPPO framework to train a policy capable of both task allocation and revision. Unlike existing approaches [12], [27], where task revisions occur only after a robot reaches its assigned task location and either completes it or fails due to the absence of a coalition, our trained policy proactively reallocates tasks as needed, based on evolving local observations and information shared by neighboring robots. As demonstrated in Section V, this integrated approach enables our framework to concurrently learn task allocation and revision, ensuring optimization of the overall objective in Dec-POMDP.

V. Evaluation

We conducted a series of simulations to evaluate: (i) the performance of our proposed framework compared to existing methods, including the state-of-the-art market-based approach PCFA,[4] (ii) its scalability with respect to the number of robots, and (iii) its generalizability across a range of task allocation scenarios with varying task difficulties. Furthermore, we present an ablation study to underscore the critical role of task revision in dynamic coalition formation.

A. Simulation Setup and Performance Metric

To evaluate our proposed method, we developed *GyMRT²A*, a Gym environment for discrete-space, discrete-time MRTA, and extended the existing MAPPO implementation from the EPyMARL [34] package[5]. Our extensions include features such as task revision, action filtering, multi-dimensional observation and action spaces, support for custom state space, and tailored actor and critic Neural Network (NN) architectures. The hyperparameters of MAPPO, presented in Table IIa, remain constant across all simulations. Each episode is limited to 100 time steps ($T = 99$) to ensure a finite episodic reward.

At the start of each episode, the initial positions of robots and tasks are randomly sampled. Table III details on the initial number of tasks per level across various task settings used in the simulations. The environment is categorized based on task distribution as either homogeneous or non-homogeneous.

In the homogeneous scenario, tasks are evenly distributed across the grid, maintaining a uniform density. Conversely, the non-homogeneous scenario features an uneven distribution of tasks, leading to variations in task density. To define non-homogeneous configurations, we adopt the *random corner patch distribution*: the grid is divided into four equal-sized corner patches, and at the start of each episode, one patch is randomly chosen to contain all tasks. Within the chosen patch, the task density is uniform, while the rest of the grid remains empty. These scenarios are used to evaluate whether the learned policy enables robots to form effective coalitions for task execution while efficiently locating tasks.

The primary performance metric is the average episodic reward. We calculate the mean and standard deviation of the episodic reward across 480 episodes: 5 different seeds with 96 evaluation episodes per seed. Each seed initializes the policy and environment with different parameters (θ, ϕ, and initial locations of robots and tasks) to ensure results are not biased by any specific initialization.

Computational Resources: The policies were trained for 4M steps over a period of up to 13 hours and up to 68 hours in the non-homogeneous and homogeneous settings, respectively. Each policy is trained using 16 CPU cores (Intel Xeon Platinum 8260) and one GPU (Nvidia V100). After training, however, each trained policy executes in just 2.1 ± 1.5 ms on a single CPU core of the same model.

B. Simulation Results

We conducted a series of simulations to evaluate the proposed framework and the baseline across various environment settings, characterized by different task spawn configurations and distributions, while keeping other parameters constant. A summary of all simulation parameters is provided in Table IIb. In the homogeneous task distribution scenario, the task spawning area is four times larger than in the non-homogeneous case. To ensure consistent robot and task densities across both distributions, the number of robots and tasks in the homogeneous scenario is proportionally increased by a factor of four. For this purpose, we use the M2 and 4×M2 settings for the non-homogeneous and homogeneous scenarios, respectively, as described in Table III.

Fig. 3 compares the performance of our model, the baseline (PCFA), and various ablation studies on the components of our model across non-homogeneous and homogeneous task distributions and five task spawn configurations (Bernoulli spawn with probabilities $p = 0.01, 0.02, 0.03, 0.04$ and instant respawn), spanning 10 different settings. Table IIc provides details of the models used in the ablation studies, highlighting their components and referencing relevant existing works

[4]The original PCFA [8] was modified to accommodate robots with limited view and communication ranges and to enable task reassignment. Details of these modifications are provided in the appendix.

[5]Our code is available at github.com/lcdbezerra/marl_mrt2a.

TABLE II

(a) MAPPO Parameters.

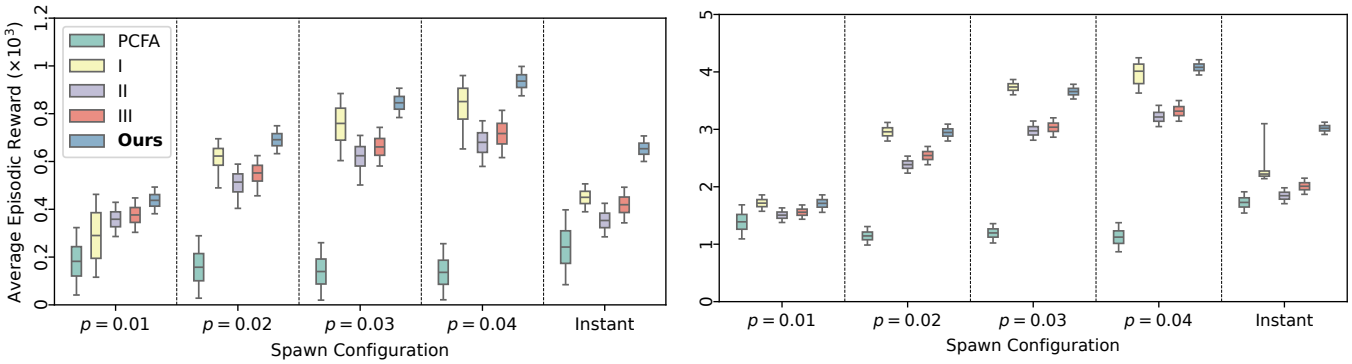
Parameter	Value	Description
η	$5 \cdot 10^{-4}$	Learning rate
γ	0.95	Discount factor
T_{rct}	5	Adv. estimation steps
ε-clip	0.2	Clipping in PPO loss
c_s	0.01	Entropy coefficient
Batch size	10	
Buffer size	10	
Reward norm.	Yes	
Return norm.	No	

(b) Default environment settings.

Parameter	Value(s)	
Grid size	20 x 20	
View range	5	
Comm. range	8	
L_{max}	3	
Task spawn	Instant respawn / Bernoulli ($p = \{0.01, \ldots, 0.04\}$)	
Task distribution	Random corner patch	Homogeneous
Number of agents	10	40
Initial task setting (see Table III)	M2	4xM2

(c) Model definitions and corresponding work.

Model	Components				Related Work
	Action Filtering	Spatial Actions	Intention Sharing	Task Revision	
PCFA	✓		✓	✓	[8]
I	✓				[10]
II	✓	✓			[11]
III	✓	✓	✓		[12]
Ours	✓	✓	✓	✓	–



(a) Non-homogeneous tasks.

(b) Homogeneous tasks.

Fig. 3: Setting-specific performance across various spawn configurations.

TABLE III: Task settings.

Setting Name	Tasks			Setting Name	Tasks		
	1	2	3		1	2	3
Easy				**Hard**			
E1	15	-	-	H1	-	5	5
E2	10	-	-	H2	-	-	10
E3	5	5	-	H3	-	-	5
Medium				**Non-Homogeneous**			
M1	6	4	4	M2	4	3	3
M2	4	3	3	**Homogeneous**			
M3	2	2	1	4×M2	16	12	12

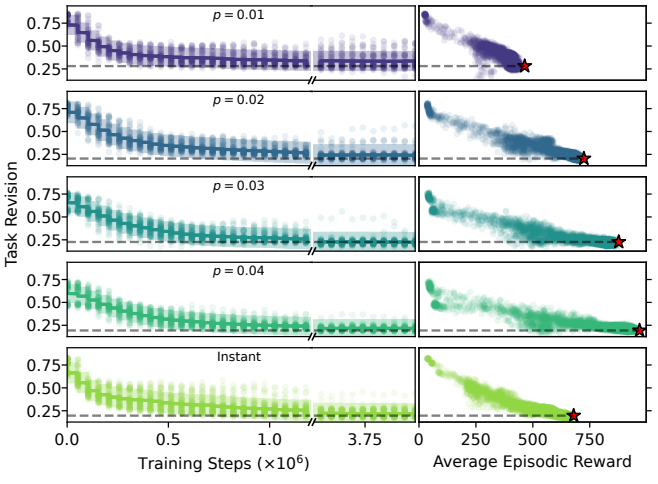


Fig. 4: The graphs illustrate how task revision rates – defined as the average number of time steps between each robot's task selection changes – evolve over the course of training, and how the average episodic reward improves as the robots learn optimal revision strategies. The dotted horizontal lines indicate the optimal revision rates.

associated with each model.[6] Each model was trained and evaluated in every environment setting.

Models I, II, and III in this study are closely related to existing works [10], [11], and [12], respectively, but include the following necessary modifications: (i) All three models are trained using MAPPO within a Dec-POMDP framework, whereas the original references rely on the MDP assumption with individual rewards. (ii) While [11] focuses on individual-robot training scenarios, model II is trained for MRTA using our framework. (iii) In [12], only level-1 tasks are considered, and a method similar to Independent Q-Learning (IQL) [40], which encourages competition rather than cooperation among multiple robots, is used, while model III uses MAPPO.

As shown in Fig. 3, our method consistently outperforms PCFA in all configurations. Notably, as the spawn rate p increases, PCFA's performance declines, indicating its difficulty in coordinating robots to effectively prioritize and allocate tasks of varying difficulty levels. Moreover, we observe that task revision significantly improves performance, particularly in highly dynamic scenarios such as the instant respawn and Bernoulli spawn with $p = 0.04$. These results align with our expectation that in environments where tasks spawn frequently and dynamically, robots must continuously reassess and update their long-term task execution plans.

While frequent adjustments to task execution plans might seem disruptive to coalition formation among multiple robots working on high-level tasks, our proposed method enables robots to learn an optimal task revision strategy, resulting in

[6] *Spatial action* refers to defining the action in Dec-POMDP as the location of a task. Consequently, in model I, which does not incorporate spatial action, an action is defined as a neighboring cell on the grid that a robot can move to within a single time step.

superior performance. Fig. 4 depicts how the robots improve their average episodic reward by learning optimal task revision strategies. Importantly, this behavior is not manually encoded – specifically, the task revision rate is not a learnable parameter – but rather emerges from the learning process within our framework. Through repeated training, the robots progressively refine their optimal task revision rates, leading to more effective coalition formation and consistent improvements in average reward.

The ablation studies evaluate the impact of intention sharing and task revision on our model's performance. Models without task revision are limited to selecting a new task only upon reaching assigned tasks, regardless of whether it executes the tasks or not. As shown in Fig. 3, model II (without intention sharing or task revision) underperforms both model III (with intention sharing only) and our model (with both intention sharing and task revision), highlighting the importance of learning the optimal timing for task revision and its communication with neighboring robots.

In the homogeneous task distribution case, model I performs comparably to our model across all Bernoulli spawn settings but underperforms in the instant respawn scenario. This outcome indicates that homogeneous task scenarios with Bernoulli spawn do not necessitate long-term planning. However, in the instant respawn scenario, the constant task density discourages simplistic approaches like those employed by model I, resulting in its reduced performance.

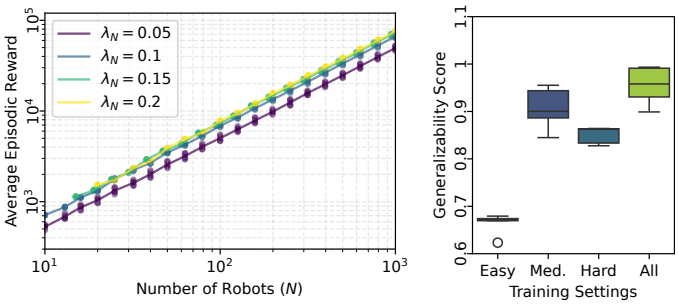
C. Scalability and Generalizability Evaluation

To evaluate scalability, we first trained policies in a homogeneous task distribution environment ($W = 10$) with a fixed number of robots. The trained policy was then tested while progressively scaling the environment size W and the number of tasks T_t to maintain constant robot density ($\lambda_N = N/W^2$) and constant task density ($\lambda_{T_t} = T_t/W^2 = 0.1$). To ensure T_t is time-invariant, we focused on the instant respawn scenario, using the M2 task setting (see Table III). Multiple robot densities were evaluated: $\lambda_N \in \{0.05, 0.1, 0.15, 0.2\}$.

Fig. 5a illustrates the performance of trained policies as a function of $N \in [10, 1000]$. We observe that the policy's performance scales linearly with N for all robot densities evaluated. These results validate the scalability of our method, demonstrating its effectiveness in managing larger numbers of robots for task allocation while maintaining constant robot and task densities.

To evaluate the generalizability of our method across environments with varying task settings, we train and evaluate our model on a diverse set of environments, each characterized by different task distributions and levels. As detailed in Table III, we designed nine task settings, categorized as easy, medium, or hard, each defined by a specific combination of task levels and quantities.

To evaluate the performance of trained policies across different task settings, we introduce the *generalizability score*. This score is calculated as the average of normalized rewards achieved by each trained policy within a specific category. The normalized reward is defined as the policy's average reward in a given setting, scaled by the maximum attainable reward. The maximum reward is estimated based on the average reward achieved by a policy trained specifically for that setting. Consequently, a policy with strong generalizability will achieve the score closer to 1.



(a) (b)

Fig. 5: Scalability (a) and generalizability (b) results.

Fig. 5b shows the generalizability scores of policies trained in easy, medium, and hard task settings, as well as in a combined setting encompassing all three. The results demonstrate that the policy trained across all settings achieves better and more consistent generalization. This policy performs close to the highest performance attainable by setting-specific policies. These results indicate that our method enables trained policies to allow robots to achieve near-optimal performance across environments with varying task-level distributions when trained in diverse environment settings.

VI. Conclusions

We introduced a learning-based framework to address decentralized dynamic coalition formation. By leveraging the MAPPO algorithm and incorporating spatial action maps, motion planning, task allocation revision, and intention sharing, our model demonstrated significant performance improvements over existing approaches. Future directions include conducting experiments with physical multi-robot systems, such as multiple mobile manipulators collaboratively transporting heavy objects, where sensor noise and communication delays may substantially impact the performance of the proposed framework. Additionally, we plan to explore fine-tuning our learning-based framework using data collected from these real-world experiments to further optimize its performance.

Appendix

Modifications to PCFA: We implemented PCFA and integrated it with *GyMRT²A* for performance comparison. To ensure a fair comparison with our method, the following modifications are applied: (i) **Utility Function**: The utility function s_k, associated with the k-th task, was adjusted to align with the reward function used in our framework: $s_k(t_{\text{ETA}}) = l_k^2\, e^{-2(t_{\text{ETA}}+t_w-t_k^0)}$, where l_k is the task level, t_{ETA} is the estimated time of arrival, t_w is the wait time, and t_k^0 is the task's initial spawn time. (ii) **Partial Observability**: In PCFA, the utility function for a task is set to zero if the task lies outside the robot's view range R_{view}, preventing robots from forming coalitions for tasks they cannot observe.

Additionally, the communication graph is updated at each time step, with links existing only between robots that are within the communication range R_{comm}. (iii) **Task Assignment Revision**: To align with our method, robots are allowed to revise their task assignments at every time step. This is achieved by recomputing all four stages of PCFA at every time step. While this modification increases communication overhead compared to the original PCFA, it allows a fair comparison in our simulation study.

REFERENCES

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] E. Tuci, M. H. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the state-of-the-art," *Frontiers Robotics AI*, vol. 5, p. 335700, 5 2018.

[3] C. Mouradian, J. Sahoo, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A coalition formation algorithm for multi-robot task allocation in large-scale natural disasters," *2017 13th International Wireless Communications and Mobile Computing Conference, IWCMC 2017*, pp. 1909–1914, 7 2017.

[4] M. S. Couceiro, D. Portugal, J. F. Ferreira, and R. P. Rocha, "Semfire: Towards a new generation of forestry maintenance multi-robot systems," *Proceedings of the 2019 IEEE/SICE International Symposium on System Integration, SII 2019*, pp. 270–276, 4 2019.

[5] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Studies in Computational Intelligence*, vol. 604, pp. 31–51, 2015.

[6] F. Quinton, C. Grand, and C. Lesire, "Market approaches to the multi-robot task allocation problem: a survey," *Journal of Intelligent & Robotic Systems*, vol. 107, 2 2023.

[7] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, pp. 912–926, 2009.

[8] G. Oh, Y. Kim, J. Ahn, and H. L. Choi, "Market-based task assignment for cooperative timing missions in dynamic environments," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 87, pp. 97–123, 7 2017.

[9] A. K. A., D. U. J., and U. Subramaniam, "A systematic literature review on multi-robot task allocation," *ACM Comput. Surv.*, vol. 57, Nov. 2024.

[10] S. Shaw, E. Wenzel, A. Walker, and G. Sartoretti, "ForMIC: Foraging via multiagent RL with implicit communication," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4877–4884, 6 2020.

[11] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser, "Spatial action maps for mobile manipulation," *Robotics: Science and Systems*, 4 2020.

[12] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, "Spatial intention maps for multi-agent mobile manipulation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 8749–8756, 3 2021.

[13] W. Weng and X. Zhu, "U-net: Convolutional networks for biomedical image segmentation," *IEEE Access*, vol. 9, pp. 16591–16603, 5 2015.

[14] X. Bai, W. Yan, S. S. Ge, and M. Cao, "An integrated multi-population genetic algorithm for multi-vehicle task assignment in a drift field," *Information Sciences*, vol. 453, pp. 227–238, 2018.

[15] X. Bai, A. Fielbaum, M. Kronmüller, L. Knoedler, 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.

[16] S. Park, Y. D. Zhong, and N. E. Leonard, "Multi-robot task allocation games in dynamically changing environments," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 13415–13422, 2021.

[17] S. Park and J. Barreiro-Gomez, "Payoff mechanism design for coordination in multi-agent task allocation games," *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 8116–8121, 2023.

[18] S. Park, "Tuning rate of strategy revision in population games," *2023 American Control Conference (ACC)*, pp. 423–428, 5 2023.

[19] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," *2019 18th European Control Conference, ECC 2019*, pp. 2071–2076, 6 2019.

[20] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 248–260, 2020.

[21] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[22] S. Singh, A. Srikanthan, V. Mallampati, and H. Ravichandar, "Concurrent Constrained Optimization of Unknown Rewards for Multi-Robot Task Allocation," in *Proceedings of Robotics: Science and Systems*, 2023.

[23] B. Xie, S. Chen, J. Chen, and L. C. Shen, "A mutual-selecting market-based mechanism for dynamic coalition formation," *International Journal of Advanced Robotic Systems*, vol. 15, 2 2018.

[24] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation," *IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 4583–4590, 8 2020.

[25] C. Sun, J. Orbik, C. M. Devin, B. H. Yang, A. Gupta, G. Berseth, and S. Levine, "Fully autonomous real-world reinforcement learning with applications to mobile manipulation," in *Conference on Robot Learning*, pp. 308–319, PMLR, 2022.

[26] A. Agrawal, A. S. Bedi, and D. Manocha, "Rtaw: An attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2023-May, pp. 1393–1399, 9 2022.

[27] A. Krnjaic, R. D. Steleac, J. D. Thomas, G. Papoudakis, L. Schäfer, A. W. K. To, K.-H. Lao, M. Cubuktepe, M. Haley, P. Börsting, and S. V. Albrecht, "Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.

[28] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024.

[29] J. Foerster, N. Nardell, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," *34th International Conference on Machine Learning, ICML 2017*, vol. 3, pp. 1879–1888, 2 2017.

[30] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 24611–24624, 2022.

[31] F. Christianos, L. Schäfer, and S. V. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 2020-December, 6 2020.

[32] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*, pp. 4295–4304, PMLR, 2018.

[33] K. Shibata, T. Jimbo, T. Odashima, K. Takeshita, and T. Matsubara, "Learning locally, communicating globally: Reinforcement learning of multi-robot task allocation for cooperative transport," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 11436–11443, 2023.

[34] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.

[35] J. Cao, L. Yuan, J. Wang, S. Zhang, C. Zhang, Y. Yu, and D. C. Zhan, "Linda: multi-agent local information decomposition for awareness of teammates," *Science China Information Sciences*, vol. 66, pp. 1–17, 8 2023.

[36] K. Boggess, S. Kraus, and L. Feng, "Toward policy explanations for multi-agent reinforcement learning," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 109–115, 4 2022.

[37] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[38] D. V. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *Journal Of Artificial Intelligence Research*, vol. 16, pp. 389–423, 6 2011.

[39] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems*, vol. 10642 LNAI, pp. 66–83, Springer, 2017.

[40] M. Tan, "Multi-agent reinforcement learning: independent versus cooperative agents," in *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, p. 330–337, 1993.