Multiagent Resource Constrained Payload Transport

Ashutosh Gupta[1], Jeremiah Goddard[1], and Mohitvishnu Gadde[1]

***Abstract*—Multiagent payload transport requires significant coordination, where each agent's performance relies on accurate observations and efficient decision-making. Full state observability and centralized control are computationally expensive and impractical in real-world settings, limiting scalability. To address these challenges, we propose a decentralized reinforcement learning (RL) framework for multiagent payload transport, where agents rely solely on local observations and decentralized control without any inter-agent communication. We model the task as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) and employ centralized training with decentralized execution (CTDE) to overcome resource constraints. Our approach enables agents to implicitly coordinate, optimize energy usage, and achieve scalable payload transport with a variable number of agents. We evaluate our method in a 2D environment with transport tasks of increasing complexity. Results demonstrate that our method scales from 2 to 50 agents without retraining and achieves comparable performance to baselines with access to global state information. These findings highlight our approach's robustness, scalability, and practicality in resource-constrained and partially observable environments.**

***Index Terms*—Multiagent transport, Reinforcement Learning, Resource constraints.**

I. Introduction

Payload transport is a fundamental task with diverse real-world applications, including construction [1], agriculture [2], search and rescue [3], and space exploration [4], [5]. To address the problem of scalability and adaptivity to varying payloads, collaborative multiagent transport systems have been explored, where multiple agents must cooperate to move and navigate the payload through complex environments [6], [7]. Decentralized systems are often preferred for scalability [8], but they require efficient management of each agent's available resources, system observability and sensing, and communication with other agents [9]–[11]. Limited communication and reliance on local sensing require each agent to adapt independently to the environment while coordinating with other agents. Designing robust systems that balance efficiency, stability, and scalability under these constraints remains a critical and open challenge in multiagent research.

Existing work on multiagent payload transport has focused on transporting a rigid structure of the payload with limited capability to handle more agents in the system [5], [12], [13]. These approaches often assume the presence of rigid, fixed-constrained joints between the agents, limiting the flexibility of the system to adapt to different payload sizes and shapes [14], [15]. Additionally, some researchers have explored the planning and coordination of non-connected multiple agents

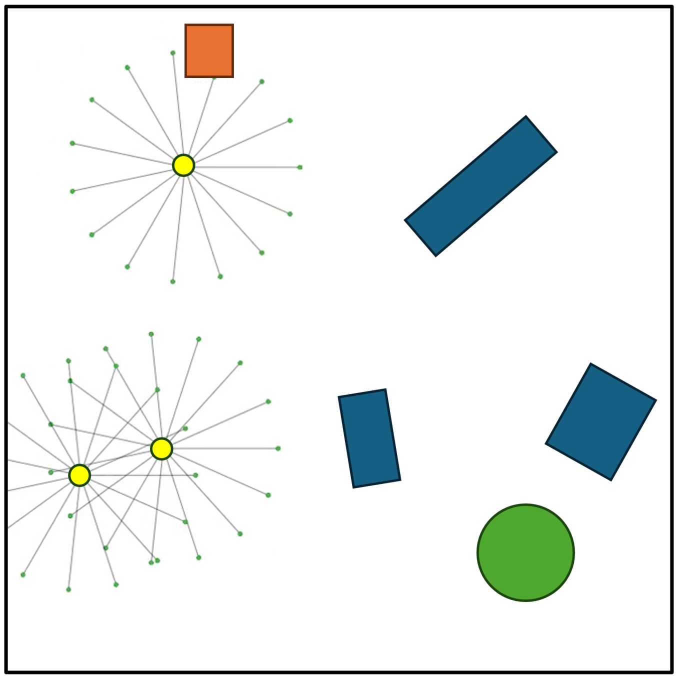


Fig. 1: An illustration of the multiagent payload transport environment: Yellow circles represent the agents, which collaboratively align to push the payload (orange square) toward a predefined goal location (green circle) while navigating around obstacles and maintaining coordination.

for collaborative transport [7], [16], [17]; they typically rely on assumptions of reliable centralized communication and no constraints on available agent-based resources. These assumptions reduce the practicality of such solutions in real-world scenarios, where communication may be unreliable, and agents operate with varying resource constraints. Currently, no approach effectively addresses the problem of planning, coordinating, and reconfiguring multiple agents for payload transport while optimizing for resource, capacity, and communication constraints. This gap highlights the need for more robust, decentralized solutions to better accommodate real-world applications' complexities.

To address these challenges, we propose a decentralized, end-to-end reinforcement learning (RL) framework for collaborative multiagent payload transport under resource-constrained, communication-free, and partially observable settings. Our approach provides technical solutions for key issues such as agent resource constraints and local system observability and promotes inter-agent coordination. Each agent is equipped with limited local sensing (e.g., lidar) and

[1] Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis, OR 97331, USA.
`{guptaash, goddarje, gaddem}@oregonstate.edu`

access only to its position, velocity, and goal information. It operates without global state knowledge or inter-agent communication. To enable effective coordination, we design an RL-based policy that leverages local sensory inputs and self-state information to generate control actions for each agent. Using an Independent Proximal Policy Optimization (IPPO) framework with centralized training and decentralized execution (CTDE), agents learn to collaboratively push the payload, avoid obstacles, and maintain coordination. Unlike prior methods, our approach scales efficiently with the number of agents and adapts dynamically to payload variations.

The primary contributions of this work are as follows:

- A reinforcement learning framework for multiagent payload transport that addresses constraints on agent resources, limited system observability, and the absence of central communication.
- A flexible policy design that adapts to varying payload weights and sizes while scaling efficiently with the number of agents without requiring policy retraining.
- An efficient learning framework that enables agents to develop robust control strategies deployable in real-world scenarios under practical constraints such as partial observability, limited sensing, the absence of centralized control, and no inter-agent communication.

The rest of the paper is organized as follows: section II discusses related work and the limitations of existing methods. section III describes our RL-based methodology, including network design, agent coordination, and the curriculum learning process. section IV outlines the experimental setup, benchmarks, and training details. section V presents our empirical results, demonstrating the performance improvements of our approach compared to various baselines, with access to global state information. section VI summarizes the proposed approach and discusses the future work.

II. Background

This section provides an overview of essential topics relevant to this work. We begin by introducing the framework of Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) for modeling decision-making under uncertainty. We then discuss the significance of multiagent systems in the context of payload transportation. Next, we examine the resource constraints that need to be addressed while developing algorithms for the deployment of multiagent systems in real-world scenarios. Finally, we explore the role of reinforcement learning in enhancing coordination and learning efficiency in multiagent systems.

A. Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs)

Various multiagent systems operate in environments where they need to make decisions based on incomplete or uncertain information. The Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) is a framework that allows cooperative behaviors within such systems to execute described goals [18].

The Dec-POMDP is an extension of the standard Markov decision process (MDP) that involves the coordination of multiple agents making decentralized decisions based on their local observations [19], [20]. Thus, agents only have partial information about the global state, and no central decision-maker provides directions for further actions. This framework holds immense significance in multiagent systems because it captures the essence of decentralized decision-making in an environment where communication is severely limited or expensive [21], [22].

In this work, we formulate the multiagent resource-constrained payload transport as a Dec-POMDP, where each agent independently learns a policy to contribute towards achieving a shared transportation goal. Formally, the Dec-POMDP can be described by the tuple $(N, S, A, O, P, R, \Omega, \gamma)$, where:

- N: Set of agents, with each agent $i \in \{1, 2, \ldots, n\}$ having partial observability over the environment.
- S: Set of global states representing the positions of agents, obstacles, and the payload within the environment.
- A: Set of joint actions, where $a = (a_1, a_2, \ldots, a_n)$ includes individual continuous actions a_i for each agent i, representing directional forces applied in the x and y dimensions to transport the payload.
- O: Set of observations for each agent, where $o_i \in O$ represents the local observation available to agent i. Observations include the agent's position, velocity, lidar readings (when enabled), relative positions to the payload and goal, and information on nearby obstacles.
- P: Transition probability function $P(s'|s, a)$, describing the probability of moving from state s to state s' given a joint action a.
- R: Reward function $R(s, a)$, providing a scalar reward based on the success of the payload transport task.
- Ω: Observation function $\Omega(o|s, a)$, describing the probability of each agent receiving an observation o given the state s and joint action a, reflecting the limited sensing capabilities and field of view for each agent.
- γ: Discount factor $\gamma \in [0, 1]$ determining the importance of future rewards, incentivizing agents to reach the goal as quickly and efficiently as possible.

B. Multiagent Systems for Payload Transportation

Efficient transportation of goods from one place to another is crucial across various industries, such as manufacturing, construction, logistics, and shipping. Traditional methods involve human-operated machinery and physical labor to complete such tasks. These approaches accomplish the task, but they may not be ideal in all situations, particularly when dealing with heavy and irregular payloads. To overcome these challenges, cooperative systems present an effective alternative to the limitations mentioned earlier. There is extensive literature on autonomous multiagent systems that collaborate to achieve tasks for transporting payloads from one location to another. [23]–[28]

This collaboration allows the division of labor within the system to lift heavier payloads that are beyond the capabilities of individual agents. Various research teams have demonstrated successful implementation of such multiagent systems on real robots to achieve cooperative payload transportation from one place to the other [6], [14], [29].

C. Resource Constraints in Multiagent Systems

Deploying a multiagent system in real-world scenarios comes with its own challenges. Since these cannot work indefinitely, we need to consider some constraints while designing algorithms to operate such systems [30], [31]. Some of these constraints include:

- **Battery life**: Limited energy supply for individual agents must be carefully considered when assigning tasks, as it directly impacts the agents' ability to complete their missions.
- **Payload capacity**: Each agent in a multiagent system has mechanical limitations that define its maximum load-carrying capacity, which must be factored into task assignments.
- **Computational resource**: Each agent requires sufficient computational resources to operate effectively, whether it follows commands from a centralized control center or operates autonomously in a decentralized manner.

Therefore, effective management of these constraints is critical for task completion in multiagent systems, as overloading an individual agent can lead to mechanical failure or premature battery depletion due to inefficient resource allocation. Thus, resource-aware task allocation and planning are essential to ensure the efficiency and reliability of such multiagent systems [32].

D. Reinforcement Learning in Multiagent Systems

Reinforcement Learning (RL) is a learning paradigm where an agent learns to make decisions through their interactions in an environment to maximize cumulative rewards [33], [34]. For multiagent systems, RL allows agents to learn optimal behaviors amidst complex environments without explicit programming [35]. Various approaches have shown that RL enables each agent to learn its policy independently, treating other agents as a part of the environment. In such setups, each agent treats other agents as part of a dynamically changing environment. These agents adapt by changing their actions in response to both the physical environment and the evolving behaviors of other agents. [36], [37] However, approaches that account for the joint action space like those discussed by Dingband Liu et. al [8], suffer from scalability problems due to the exponential growth of the joint action space.

Centralized Training with Decentralized Execution (CTDE) [38] is emerging as one of the potential methods to train multiagent systems, which are trained using centralized information but execute the learned policies based on local observation of individual agents. These approaches have demonstrated promising results, exhibiting improved coordination and enhanced scalability during execution [39], [40].

Multiagent actor-critic methods, such as Proximal Policy Optimization (PPO) [41], effectively stabilize learning in multiagent settings by having the actor use the value function generated by the learned critic to guide policy updates, reducing the effects of non-stationarity [42], [43]. When combined with a recurrent neural network (RNN), PPO can handle partial observability by maintaining an internal state that encodes historical information for decision-making [44], [45]. However, while PPO is highly efficient, applying it across multiple agents remains computationally demanding.

III. Methodology

We formulate the multiagent resource-constrained payload transport as a Decentralized Partially Observable MDP (Dec-POMDP) [18]. We consider that each agent only has access to the static global goal position, its own state, and sensor measurements without access to the complete state information of the system. With only partial observability, each agent independently contributes to the payload transport task by directly learning the low-level actions to push the package to the desired goal. Our approach employs Independent Proximal Policy Optimization (IPPO) [46] with Centralized Training and Decentralized Execution (CTDE). Below, we expand on the policy architecture, training environment, reward function, and training details.

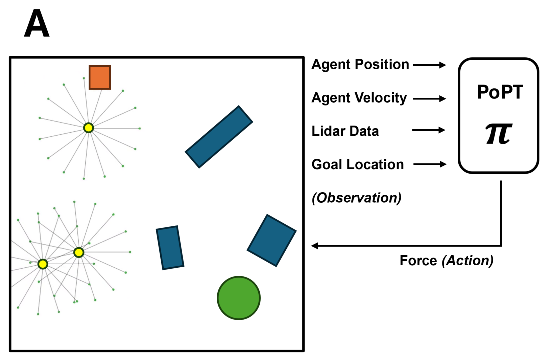
A. Policy Architecture

The policy and the critic networks are composed of MLPs with hidden layer dimensions as $\{256, 256\}$, and the activation function for all layers is $tanh$ [47]. We use a shared $Adam$ optimizer [48] for both the policy and the critic. The policy input includes the static global goal position $p_g = \{x_g, y_g\}$, agent's local observations include its current planar position $p_a = \{x_a, y_a\}$, current planar velocity $v_a = \{\dot{x}_a, \dot{y}_a\}$, and lidar sensor measurements. Each agent is equipped with a simulated lidar sensor with 72 rays, which provides a 5° resolution and a maximum range of $2m$ to sense the environment. The total input size to the policy network is 78. The policy outputs a 2-dimensional directional force vector $a = \{f_x, f_y\}$ to control the agent's movement in the x and y directions.

B. Episode Generation

Our approach employs a modified version of IPPO, which generates training data from episodes involving three agents, as described in Algorithm 1. During each episode, transitions from all agents are aggregated into a shared replay buffer to optimize a single actor and a centralized critic using the PPO objective [41]. The centralized critic provides a global perspective to improve policy updates, accelerating convergence to an optimal policy. Since all agents are homogeneous, we share the actor and critic parameters across agents, enabling more effective policy learning from collective experiences. After training, the shared actor is deployed as the control policy for all agents during execution, as outlined in Algorithm 2.

At the start of each training episode, three agents are randomly spawned near the left end of the environment,



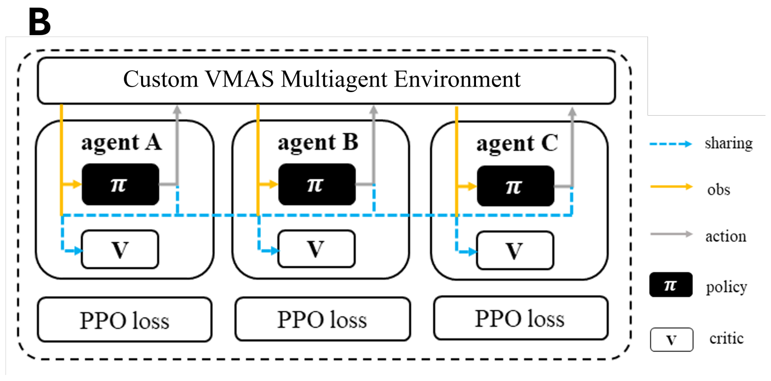


Fig. 2: **[A]** Illustrates the Partially Observable Payload Transport (PoPT) policy, a single controller that can be deployed to any number of agents for pushing a payload to a goal position without requiring inter-agent communication. **[B]** The framework of Independent Proximal Policy Optimization (IPPO) with Centralized Training and Decentralized Execution (CTDE) is used for policy training.

while the goal location is randomized near the right end. The package is initialized in the middle region of the environment with a constant mass, size, and orientation throughout training. To introduce variability, each agent's pushing capacity is randomized at the start of each episode, within the range of $capacity = [0.8, 1.2]$. Each episode terminates after 500 time steps or when the package reaches the goal location.

Algorithm 1 Training Phase: Partially observable Payload Transport (PoPT) with CTDE and IPPO

```
Initialize: Shared actor-critic network parameters $\theta$ and $\phi$
Input: Package mass and size, environment env, max. agents $n_{max}$, number of episodes $T$, curriculum levels
for each iteration in total iterations do
    Reset env and initialize agents and goal position
    for each time step $t = 1$ to $T$ do
        for each agent $i = 1$ to $n$ (in parallel) do
            Sample action $a_i^t \sim \pi_\theta(a_i|o_i^t)$ based on current observation $o_i^t$
            Execute $a_i^t$, applying a fixed proportion of $a_i^t$ as force on the package
            Observe reward $r_i^t$ and new observation $o_i^{t+1}$
            Store $(o_i^t, a_i^t, r_i^t, o_i^{t+1})$ for training
        end for
    end for
    Update $\theta$ and $\phi$ with IPPO using experience from all agents
end for
return Trained policy $\pi_\theta$ for deployment
```

C. Reward Design

The proposed reward function consists of four key components:

1) **Global Reward:** A terminal reward given when the package reaches the goal position.

Algorithm 2 Execution Phase: Rule-Based Heuristic Deployment of Multiagent System

```
Input: Package mass and size, environment env, maximum number of available agents $n_{max}$
Initialize scenario:
Determine the optimal number of agents $n \leq n_{max}$ based on package mass and size (rule-based heuristic)
Evenly space $n$ agents on the leftmost portion of env
Randomly place obstacles in the middle region, ensuring minimum spacing of $2\times$ package size
Set the goal position within the rightmost portion of env
for each time step $t$ do
    for each agent $i = 1$ to $n$ (in parallel) do
        Execute action $a_i^t \sim \pi_\theta(a_i|o_i^t)$ learned during training
    end for
    Observe environment state, rewards, and agent positions
end for
```

2) **Goal Shaping Reward:** A dense reward based on the Euclidean distance moved by the package toward the goal at each time step, encouraging continuous progress.
3) **Package Shaping Reward:** A dense reward based on the Euclidean distance each agent moves toward the package, promoting agent coordination to reach and push the package.
4) **Energy Penalty:** A penalty computed as the sum of the norm of the agent actions at each time step, encouraging energy-efficient behavior and minimizing unnecessary movement.

These components collectively guide the agents to push the package efficiently while balancing energy consumption, coordination, and task completion. Each reward component plays a critical role in shaping agent behavior, encouraging efficient movement, collaborative effort, and energy conservation. A detailed description of all the reward components and the corresponding equations for each reward term are provided below:

$$\boldsymbol{r}_t^i = \boldsymbol{r}_{\textbf{global}} + \boldsymbol{r}_{\textbf{goal\_shaping}} + \boldsymbol{r}^i_{\textbf{package\_shaping}} - \boldsymbol{r}_{\textbf{energy}} \tag{1}$$

- r_{global} Global reward:

$$\boldsymbol{r}_{\textbf{global}} = \boldsymbol{w}_g \cdot \mathbb{1}_{\textbf{on\_goal}}$$

  - w_g: Reward for reaching the goal $w_g = 100$.
  - $\mathbb{1}_{\text{on\_goal}}$: Indicator function set to 1 when the package reaches the goal.

- $r_{\text{goal\_shaping}}$ Goal shaping reward:

$$\boldsymbol{r}_{\textbf{goal\_shaping}} = \overline{\boldsymbol{r}}_{\textbf{goal\_shaping}} - (\boldsymbol{w}_{\textbf{gs}} \cdot ||\boldsymbol{p}_{\textbf{goal}} - \boldsymbol{p}_{\textbf{package}}||)$$

  - w_{gs}: Goal shaping reward weight $w_{\text{gs}} = 50$.
  - $\overline{r}_{\text{goal\_shaping}}$: Goal shaping component from previous time step.
  - $||p_{\text{goal}} - p_{\text{package}}||$: Current Euclidean distance between the goal and the package.

- $r^i_{\text{package\_shaping}}$ Package shaping reward for each agent:

$$\boldsymbol{r}^i_{\textbf{package\_shaping}} = \overline{\boldsymbol{r}}^i_{\textbf{package\_shaping}} - (\boldsymbol{w}_{\textbf{ps}} \cdot ||\boldsymbol{p}_{\textbf{package}} - \boldsymbol{p}^i_{\textbf{agent}}||)$$

  - w_{ps}: Package shaping reward weight $w_{\text{ps}} = 20$.
  - $\overline{r}^i_{\text{package\_shaping}}$: Package shaping component for the i-th agent from previous time step.
  - $||p_{\text{package}} - p^i_{\text{agent}}||$: Current Euclidean distance between the package and the i-th agent.

- r_{energy} Energy penalty:

$$\boldsymbol{r}_{\textbf{energy}} = \boldsymbol{w}_{\textbf{energy}} \cdot \sum_i^a \left(\frac{||\text{action}^i||}{\sqrt{2 \cdot (\text{action}_{\textbf{range}} \cdot \text{capacity}^i)^2}} \right)$$

  - w_{energy}: Energy penalty weight $w_{\text{energy}} = 0.1$
  - Penalty is computed as the sum of normalized norm of action for all agents.

D. Training Details

During training, we use a system with three agents, and the package mass is set to ensure that at least three agents are required to effectively push the package. A buffer of 100k transitions is collected at each iteration, and optimization is performed over 5 epochs. Training is run for a maximum of 50 iterations to ensure sufficient policy convergence. All training parameters are listed in Table I. Further details on our approach's experimental setup and validation, Partially Observable Payload Transport (PoPT), are provided in section IV.

IV. Experiment

We conduct three sets of experiments to evaluate our proposed approach against baseline methods. The evaluation focuses on three key aspects: (1) the scalability of our approach with varying numbers of agents, (2) the impact of state space observability on transport task performance, and (3) the contribution of each reward term in our formulation. To assess scalability, we conduct trials with different agent counts and pushing capacities, analyzing how our approach adapts to larger teams of agents. We also examine learning efficiency and convergence times when using only specific components of the reward function. The following subsections provide details on the training and evaluation environment, the baselines used for observability experiments, and the reward formulation ablations. In section V, we present the results for each set of experiments, highlighting the performance of our approach relative to the baselines.

Parameter	**Value**
Learning Rate (Actor & Critic)	3e-4
PPO Clip Range	0.2
Number of Epochs	5
Mini-Batch Size	256
Discount Factor (Gamma)	0.99
GAE Lambda	0.9
Value Function Coefficient	0.5
Entropy Coefficient	1e-4
Max Grad Norm	1.0
Buffer Size	100000
Total iterations	50
Episode length	500
Number of agents in training	3

TABLE I: Training parameters

A. Payload Transportation Multiagent Domain

For our experiments, we utilize the Vectorized Multi-Agent Simulator (VMAS) [49] due to its adaptability and resource efficiency. The simulator is configured to create a 2D environment where agents are tasked with pushing a package to a predefined goal position, as illustrated in Figure 1.

The number of agents in each scenario is determined using a heuristic policy that calculates the optimal number of agents required based on the package's mass and the agents' pushing capacity. At the start of each episode, agents are initialized at random positions within the leftmost region of the environment, while the package is placed randomly in the middle, and the goal is randomly positioned in the rightmost region. The agent's pushing power and spawn locations are randomized for each episode to promote robustness and generalization.

This work focuses on developing an end-to-end control policy for the agents. To facilitate this, a heuristic policy as described in Algorithm 2 is implemented as a rule-based system with sufficient information to select the required number of agents. While this approach effectively supports the objectives of this study, future work could explore more autonomous and data-driven methods to dynamically determine the number of agents.

B. Evaluating scalability

We evaluate the scalability of our proposed approach with varying numbers of agents in the system. We use three agents for training to learn individual behaviors required to push the package towards the goal. Since our approach relies solely on local agent observations, we design a policy trained with a minimal number of agents to seamlessly scale to larger agent teams without retraining. To evaluate this, we test the policy with an unseen range of 2 to 50 agents during rollout. This zero-shot generalization experiment aims to demonstrate the scalability and robustness of our approach in handling variable team sizes for the payload transport task.

C. Evaluating observability

We evaluate the impact of different state observability configurations on the performance of the approach. To benchmark our method, we define three baselines for this set of experiments. Each baseline is trained using the same reward function described in Equation 1. This evaluation highlights the role of state observability in shaping the agent's behavior and task performance.

1) ***Full State Observability:*** In this configuration, each agent has complete knowledge of the environment. In addition to its own state and velocity, each agent has access to the position and velocity of all the other agents and the static global position of the goal. Agents also have continuous access to the package state, which includes the package's relative position with respect to the goal and the agent, along with the current package velocity. Unlike other configurations, lidar measurements are excluded since the agents already have access to ground-truth information, making lidar data redundant.

This setup represents a fully observable environment where no information is hidden from the agents. While this configuration is theoretically optimal for agent decision-making, it comes with significant drawbacks. Full observability requires access to the complete environment state, leading to an exponentially larger state space as the number of agents increases. This results in higher computational complexity, as the policy is trained on a fixed observation size. Scaling to larger teams would require padding the observations, leading to information loss and inefficiencies. Consequently, separate policies must be trained for different agent counts, limiting the scalability of the approach. Policies trained for a specific number of agents may not generalize effectively to scenarios with varying team sizes.

2) ***Package State:*** In this configuration, each agent has access to only its own state and velocity without knowledge of the states of other agents. Additionally, each agent receives the static global position of the goal and continuous access to the ground-truth package state. The package state includes the relative position of the package with respect to the goal and the agent, along with the current package velocity. Similar to the previous baseline, lidar measurements are excluded due to redundancy, as agents already have access to complete information about the package.

This configuration enables scalability with respect to the number of agents since agents do not depend on the states of other agents. However, its reliance on ground-truth package states poses a significant challenge for real-world deployment, where precise and continuous access to the package's state may not be feasible. This limitation would make it difficult to generalize the approach to real-world environments with limited sensing and partial observability.

3) ***No Goal pose:*** In this configuration, each agent's observations include its own state and velocity, along with lidar measurements, consistent with our proposed approach. However, the static global goal position is intentionally excluded from the agent's observations. In this environment, the lidar sensor is only obstructed by the package and other agents, meaning the agent cannot directly sense the goal position.

This setup is designed to evaluate the importance of goal position information in our proposed method. We can assess how well agents can coordinate and push the package without explicit goal awareness by removing access to the goal. This setup highlights the role of goal position information in enabling effective coordination and goal-directed behavior.

D. Ablating reward shaping

We analyze the impact of different reward components from our formulation by performing an ablation study to demonstrate the contribution of each term to task learning. To achieve this, we define three baselines that are compared against our full method, which incorporates all four reward components. Each baseline is trained using the observation space defined by our approach and includes the energy penalty term. The remaining three reward components are ablated in different configurations to evaluate their individual impact on learning performance. This analysis provides insight into the role each reward term plays in shaping agent behavior [50] and achieving efficient task completion.

1) ***Global:*** In this baseline, only the *global reward* and *energy penalty* are utilized. The global reward is a sparse signal awarded solely when the package successfully reaches the goal. This configuration represents a minimal reward setup, relying exclusively on final task completion to guide learning. We hypothesize that the absence of intermediate feedback during the episode hinders the agents' ability to learn effective behaviors, often necessitating extensive exploration to achieve meaningful progress.

2) ***Goal shaping:*** In this baseline, the reward function includes the *global reward, energy penalty,* and *goal shaping reward*. The goal-shaping reward provides dense feedback at every time step by rewarding agents based on the distance the package moves toward the goal. Unlike the sparse global

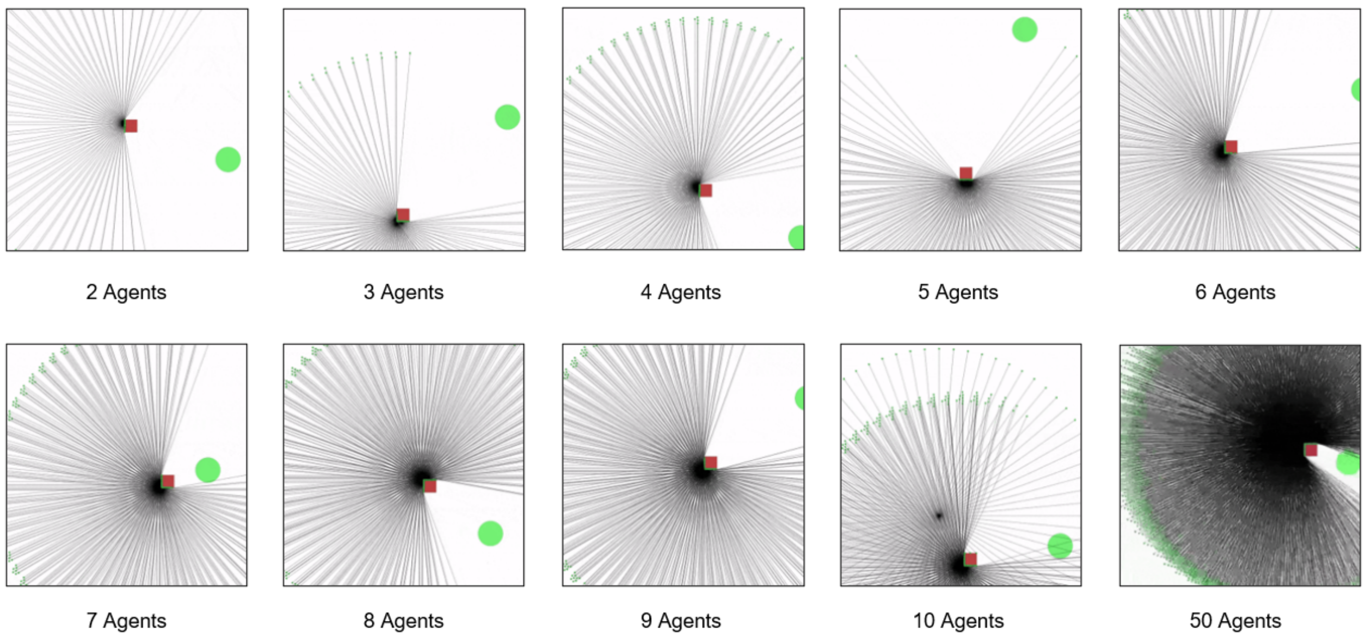


Fig. 3: Snapshot of results demonstrating the deployment of the proposed PoPT policy, trained on only 3 agents, to an unseen range of 2 – 50 agents during rollout. Since the method relies solely on local observations, it can be seamlessly deployed across any number of agents. The results showcase up to 50 agents collaboratively pushing the payload toward a predefined goal position.

reward, the goal-shaping reward offers continuous feedback throughout the episode, enabling agents to learn more efficient behaviors. We posit that this configuration helps agents develop goal-directed behavior.

*3) **Package shaping**:* In this baseline, the reward function includes the *global reward, energy penalty,* and *package shaping reward*. The package shaping reward provides dense feedback at every time step, encouraging each agent to move toward the package. Unlike the global reward, which is sparse, this shaping reward offers continuous feedback to agents throughout the episode. We expect this configuration to promote effective agent-to-package coordination.

V. Results

This section presents the performance evaluation of the proposed decentralized reinforcement learning approach for multiagent payload transport (PoPT). The evaluation is conducted in a 2D environment using the set of experiments described above, highlighting the effectiveness of our approach in terms of scalability, observability, and the impact of reward components on task learning and agent coordination.

A. Scalability results

Figure 3 presents snapshots of our policy deployed with varying numbers of agents in the system, showcasing its scalability. We deploy the policy with up to 50 agents, demonstrating their ability to push the package to the goal position collaboratively. As the number of agents increases, the time taken to complete the task decreases, but with diminishing returns due to the fixed mass and dimensions of the package.

The agents exhibit emergent coordination behavior, implicitly learning to position themselves at optimal pushing points around the package. With only 2 agents, the combined pushing power is insufficient for direct movement, often resulting in an L-shaped path to the goal. However, with 3 or more agents, they naturally distribute along the package's edges and push it in a more direct, diagonal path toward the goal, significantly improving task efficiency. This behavior highlights the adaptability and scalability of our approach, which achieves optimal coordination without requiring additional training as the number of agents increases.

B. Observability results

Figure 5 illustrates the training curves for the observability experiment, comparing the proposed method with the three baseline approaches. The results demonstrate that our method achieves performance comparable to the Full State and Package State baselines, both of which have significantly higher state space observability. In contrast, the No Goal Pose baseline fails to achieve the task, as reflected by the low rewards and flat training curve. Figure 4 provides snapshots of agent behavior during deployment for each of the four methods, highlighting differences in coordination and task execution.

The *full state baseline* grants each agent access to the complete state space, including the position and velocity of all agents as well as the package state. This extensive information enables agents to complete the task with ease. However, this method suffers from significant scalability challenges. As the

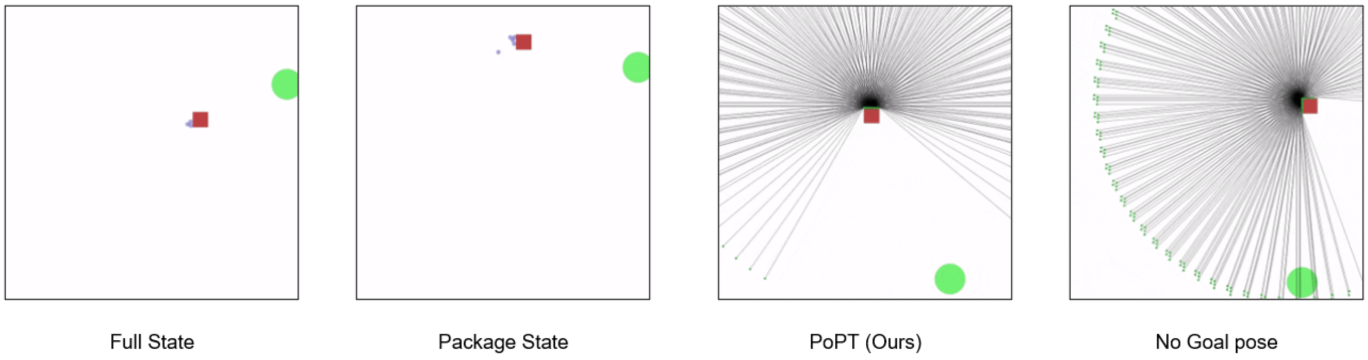


Fig. 4: Snapshot results of baseline deployments evaluating performance under different state observability conditions: **Full state:** Trained on 3 agents with access to complete state information, achieving the task but limited to 3-agent deployment due to the expanding state space. **Package state:** Accesses the continuous ground-truth state of the package, enabling successful task completion with varying agent counts. **PoPT (ours):** Relies solely on local observations, achieving comparable performance to baselines with much better state information (full state and package state). **No Goal pose:** Lacks access to the global goal position, managing to push the payload but struggling to locate the goal precisely.

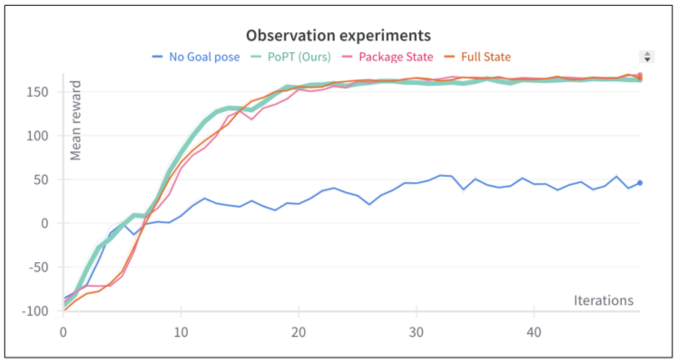


Fig. 5: Training curves of various observability configurations.

number of agents increases, the input size to the policy grows exponentially, increasing computational complexity. Additionally, since the policy is trained on a fixed number of agents (3 in this case), it is unable to generalize to varying numbers of agents. Deploying this policy on a different number of agents requires retraining from scratch, thereby compromising scalability.

The *package state baseline* achieves task completion comparable to the Full State baseline but with better scalability. This is achieved as each agent has access to the global state of the package, including its relative position and velocity to the package. However, agents do not have access to the states of other agents. While this approach scales to larger teams of agents, its reliance on the package state presents a major limitation for real-world deployment. Accessing the global package state in physical environments requires motion capture systems or external tracking setups [7], which are expensive, labor-intensive, and impractical for large-scale deployment.

The *no goal pose baseline*, the agents lack access to the static global goal position, relying solely on local state and lidar measurements. Without access to goal position information, agents exhibit poor coordination, as reflected by the low reward observed in Figure 5. Agents tend to approach the package and push it in random directions, relying on exploration to locate the goal. This limitation is also due to the lidar sensors' inability to detect the goal. Thus, leading to inefficient behavior; which highlights the importance of goal position information for effective planning and execution. We note that in real-world applications, the global goal position can be defined by the user or inferred from a high-level planner with access to an environment map, making it a practical assumption for real-world deployment.

Finally, our method with the proposed observation space achieves task success comparable to the Full State and Package State baselines but with significantly fewer assumptions about state observability. Here, each agent relies only on its local state, lidar measurements, and static goal position. Agents independently learn to approach the package and push it toward the goal, exhibiting emergent coordination. This configuration avoids dependence on ground-truth package states and agent-to-agent state sharing, making it more scalable and realistic for real-world deployment. Unlike the Full State baseline, our method generalizes to varying numbers of agents, enabling deployment without retraining. This scalability is achieved through the use of decentralized execution, where each agent operates using its own local observation.

C. Reward ablation results

Figure 7 shows the training curves for the ablation study on reward components, comparing the performance of our method with all four reward components against baselines with ablated reward terms. The results show that only our method successfully learns the task, while the other baselines struggle to learn meaningful behavior. The learned behaviors for each method are visualized in Figure 6, which shows snapshots from the deployment of all four methods.

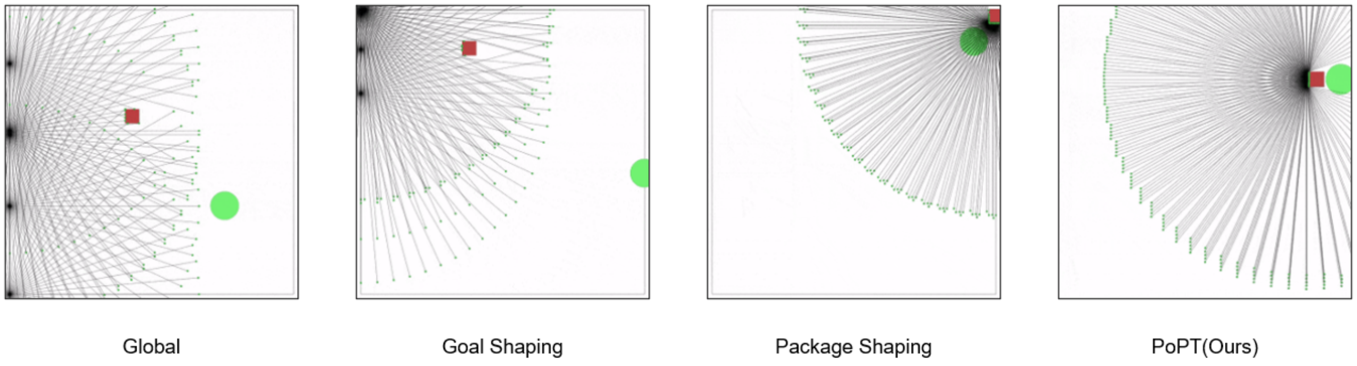


Fig. 6: Snapshot results of baseline deployments to ablate the reward components. **Global:** Utilizes only the sparse global reward and energy penalty. This baseline fails to achieve the task as the sparse reward provides insufficient feedback for the policy to learn meaningful behavior. **Goal shaping:** Incorporates goal-shaping dense rewards, global reward, and energy penalty. This baseline fails, as agents receive no feedback until they accidentally move the package, requiring extensive exploration to learn effective behaviors. **Package shaping:** Uses package-shaping dense rewards, global reward, and energy penalty. This baseline achieves partial success, as agents quickly learn to reach the package but continue to push the package randomly due to a lack of intermediate incentive to reach the goal location. **PoPT (ours):** Combines all four reward components, enabling the policy to achieve the task optimally, demonstrating the effectiveness of the proposed reward function.

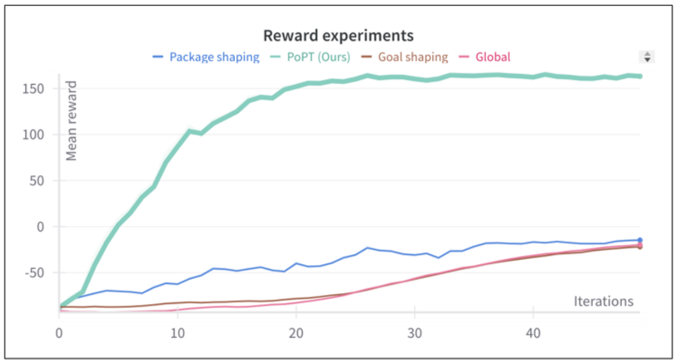


Fig. 7: Training curves for different reward configurations used in the reward ablation study.

The *Global Reward baseline* uses only the sparse global reward, provided when the package reaches the goal, along with the energy penalty. The extreme sparsity of this reward makes it difficult for the policy to explore and learn effective behaviors. Due to the constant energy penalty, agents learn to minimize energy consumption by remaining stationary throughout the episode. This demonstrates the need for dense feedback during the episode to incentivize movement and exploration. Without such feedback, the policy fails to learn any useful behavior.

The *Goal-Shaping baseline* introduces a denser feedback signal, rewarding agents for moving the package closer to the goal at every time step. While this approach offers more frequent feedback than the sparse global reward, it still struggles with exploration. The policy receives no reward until an agent accidentally makes contact with the package and moves it, leading to a similar exploration issue as the global reward baseline. As a result, the agents learn to remain stationary to avoid energy penalties. This highlights the need for an additional dense component to encourage agents to approach the package before attempting to push it.

The *Package-Shaping baseline* incorporates a dense reward encouraging agents to approach the package. This additional reward signal addresses the exploration issue seen in previous baselines, as agents now receive continuous feedback when moving toward the package. As a result, the policy learns to approach the package more effectively and initiate pushing behavior. However, without the goal shaping reward, the agents have no dense feedback after reaching the package. Consequently, the agents push the package toward the general goal region but resort to random movements in an attempt to “stumble” upon the goal. This behavior demonstrates the necessity of both goal-shaping and package-shaping rewards to achieve optimal behavior, as one reward alone is insufficient to guide the policy through both stages of the task.

Our proposed method *(PoPT)* incorporates all four reward components, providing dense feedback at every stage of the task. The package-shaping reward encourages agents to approach the package, while the goal-shaping reward guides them to push it toward the goal. The global reward reinforces task completion, and the energy penalty promotes efficient behavior. With such dense feedback present throughout the episode, the policy quickly converges to the desired behavior, with agents efficiently coordinating to approach and push the package directly to the goal. The training curve for this method shows the fastest convergence and the highest overall performance.

VI. Conclusion and Future Work

In this paper, we addressed the problem of multiagent resource-constrained payload transport by proposing a decentralized end-to-end reinforcement learning (RL) approach.

Unlike previous methods that rely on centralized control, extensive communication, and enhanced state observability, our approach leverages only local agent observations, decentralized control, and no inter-agent communication. This design enables a scalable solution for payload transport, allowing agents to implicitly learn coordination and optimize system-wide energy consumption under resource constraints.

Despite the promising results, our proposed method has certain limitations. We employed a simple rule-based heuristic policy to determine the number of agents required for each task. While sufficient for this study, this heuristic policy was not our research's primary focus. Additionally, our experiments were conducted in a 2D simulation environment with simplified physics, which does not fully capture the complexity of real-world interactions. The package properties, such as shape, dimensions, and mass, were kept constant throughout training, limiting the generalization of the learned policy to more diverse payload configurations. Finally, the training environment did not account for obstacles, which is critical for real-world tasks as agents must navigate environments with dynamic and static obstacles to ensure efficient payload delivery.

As part of future work, we aim to address these limitations and further extend the proposed method. First, the heuristic policy for agent selection could be replaced with a more autonomous and data-driven approach that selects the optimal number of agents based on dynamic environmental factors, such as package mass, size, and agent capacity. Additionally, we plan to use higher-fidelity simulators with more realistic physics to better capture the complexities of real-world interactions. To improve generalization, we intend to train the policy on a broader range of package properties, including diverse shapes, sizes, and masses. We also plan to introduce obstacles into the training environment, allowing agents to learn to navigate around them while minimizing energy consumption. Lastly, to facilitate real-world deployment, our method could be extended to incorporate robot-specific dynamics. This could be achieved through hierarchical control, where the learned policy guides a low-level robot controller, or through end-to-end training, where the policy directly controls the robot's actions. These enhancements would make our method more robust, scalable, and suitable for deployment in real-world, multi-robot payload transport scenarios.

References

[1] S. A. Prieto, N. Giakoumidis, and B. García de Soto, "Multiagent robotic systems and exploration algorithms: Applications for data collection in construction sites," *Journal of Field Robotics*, vol. 41, no. 4, pp. 1187–1203, 2024.

[2] K. Ankit, S. N. Kolathaya, and D. Ghose, "Multi-agent collaborative framework for automated agriculture," in *2021 15th International Conference on Advanced Computing and Applications (ACOMP)*, pp. 30–37, IEEE, 2021.

[3] N. Mohanty, M. S. Gadde, S. Sundaram, N. Sundararajan, and P. Sujit, "Context-aware deep q-network for decentralized cooperative reconnaissance by a robotic swarm," *arXiv preprint arXiv:2001.11710*, 2020.

[4] L. Zhang, H. Xiong, O. Ma, and Z. Wang, "Multi-robot cooperative object transportation using decentralized deep reinforcement learning," *arXiv preprint arXiv:2007.09243*, 2020.

[5] G. Eoh and T.-H. Park, "Cooperative Object Transportation Using Curriculum-Based Deep Reinforcement Learning," *Sensors*, vol. 21, p. 4780, Jan. 2021. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[6] B. Pandit, A. Gupta, M. S. Gadde, A. Johnson, A. K. Shrestha, H. Duan, J. Dao, and A. Fern, "Learning Decentralized Multi-Biped Control for Payload Transport," June 2024. arXiv:2406.17279.

[7] Y. Feng, C. Hong, Y. Niu, S. Liu, Y. Yang, W. Yu, T. Zhang, J. Tan, and D. Zhao, "Learning Multi-Agent Loco-Manipulation for Long-Horizon Quadrupedal Pushing," Nov. 2024. arXiv:2411.07104.

[8] D. Liu, F. Ren, J. Yan, G. Su, W. Gu, and S. Kato, "Scaling Up Multi-Agent Reinforcement Learning: An Extensive Survey on Scalability Issues," *IEEE Access*, vol. 12, pp. 94610–94631, 2024. Conference Name: IEEE Access.

[9] P. Agrawal, P. Varakantham, and W. Yeoh, "Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, (New York, New York, USA), pp. 10–16, AAAI Press, July 2016.

[10] F. d. Nijs, E. Walraven, M. D. Weerdt, and M. Spaan, "Constrained Multiagent Markov Decision Processes: a Taxonomy of Problems and Algorithms," *Journal of Artificial Intelligence Research*, vol. 70, pp. 955–1001, Mar. 2021.

[11] J. Cui, Y. Liu, and A. Nallanathan, "Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 729–743, Feb. 2020.

[12] J. Kim, R. T. Fawcett, V. R. Kamidi, A. D. Ames, and K. A. Hamed, "Layered Control for Cooperative Locomotion of Two Quadrupedal Robots: Centralized and Distributed Approaches," *IEEE Transactions on Robotics*, vol. 39, pp. 4728–4748, Dec. 2023. Conference Name: IEEE Transactions on Robotics.

[13] M. L. Elwin, B. Strong, R. A. Freeman, and K. M. Lynch, "Human-Multirobot Collaborative Mobile Manipulation: the Omnid Mocobots," June 2022.

[14] Q. Liu, Z. Nie, Z. Gong, and X.-J. Liu, "An Omnidirectional Transportation System With High Terrain Adaptability and Flexible Configurations Using Multiple Nonholonomic Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 8, pp. 6060–6067, Sept. 2023. Conference Name: IEEE Robotics and Automation Letters.

[15] J. Kim, J. Lee, and A. D. Ames, "Safety-Critical Coordination for Cooperative Legged Locomotion via Control Barrier Functions," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2368–2375, Oct. 2023. ISSN: 2153-0866.

[16] L. Pei, J. Lin, Z. Han, L. Quan, Y. Cao, C. Xu, and F. Gao, "Collaborative Planning for Catching and Transporting Objects in Unstructured Environments," *IEEE Robotics and Automation Letters*, vol. 9, pp. 1098–1105, Feb. 2024. Conference Name: IEEE Robotics and Automation Letters.

[17] J. Horyna, T. Baca, and M. Saska, "Autonomous Collaborative Transport of a Beam-Type Payload by a Pair of Multi-rotor Helicopters," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1139–1147, June 2021. ISSN: 2575-7296.

[18] F. A. Oliehoek, "Decentralized pomdps," in *Reinforcement learning: state-of-the-art*, pp. 471–503, Springer, 2012.

[19] C. Amato, G. Chowdhary, and A. Geramifard, "Decentralized Control of Partially Observable Markov Decision Processes,"

[20] D. S. Bernstein, S. Zilberstein, and N. Immerman, "The Complexity of Decentralized Control of Markov Decision Processes," Jan. 2013. arXiv:1301.3836.

[21] F. S. Melo, M. T. J. Spaan, and S. J. Witwicki, "QueryPOMDP: POMDP-Based Communication in Multiagent Systems," in *Multi-Agent Systems* (M. Cossentino, M. Kaisers, K. Tuyls, and G. Weiss, eds.), (Berlin, Heidelberg), pp. 189–204, Springer, 2012.

[22] F. A. Oliehoek, M. T. Spaan, N. Vlassis, *et al.*, "Dec-pomdps with delayed communication," in *The 2nd Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains*, Citeseer, 2007.

[23] H. Ma, C. Tovey, G. Sharon, T. K. S. Kumar, and S. Koenig, "Multi-Agent Path Finding with Payload Transfers and the Package-Exchange Robot-Routing Problem,"

[24] Y. Sirineni, P. Verma, and K. Karlapalem, "Traffic Management Strategies for Multi-Robotic Rigid Payload Transport Systems: Extended Abstract," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 225–227, Aug. 2019.

[25] H. Rastgoftar, J.-B. Jeannin, and E. Atkins, "Formal Specification of Continuum Deformation Coordination," in *2019 American Control Conference (ACC)*, pp. 3358–3363, July 2019. ISSN: 2378-5861.

[26] H. Rastgoftar and E. M. Atkins, "Cooperative aerial lift and manipulation (CALM)," *Aerospace Science and Technology*, vol. 82-83, pp. 105–118, Nov. 2018.

[27] H. Rastgoftar and E. M. Atkins, "Continuum Deformation of a Multiple Quadcopter Payload Delivery Team without Inter-Agent Communication," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 539–548, June 2018. ISSN: 2575-7296.

[28] H. Rastgoftar and E. M. Atkins, "Cooperative Aerial Payload Transport Guided by an In Situ Human Supervisor," *IEEE Transactions on Control Systems Technology*, vol. 27, pp. 1452–1467, July 2019. Conference Name: IEEE Transactions on Control Systems Technology.

[29] H. Zhu, S. Yang, W. Wang, X. He, and N. Ding, "Cooperative transportation of tether-suspended payload via quadruped robots based on deep reinforcement learning," in *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1–6, Dec. 2023.

[30] P. Pezeshkpour, E. Kandogan, N. Bhutani, S. Rahman, T. Mitchell, and E. Hruschka, "Reasoning Capacity in Multi-Agent Systems: Limitations, Challenges and Human-Centered Solutions," Feb. 2024. arXiv:2402.01108.

[31] C. Tong, A. Harwood, M. A. Rodriguez, and R. O. Sinnott, "An Energy-aware and Fault-tolerant Deep Reinforcement Learning based approach for Multi-agent Patrolling Problems," June 2023. arXiv:2212.08230.

[32] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, pp. 1495–1512, Oct. 2013. Publisher: SAGE Publications Ltd STM.

[33] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, May 1996.

[34] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, pp. 26–38, Nov. 2017. Conference Name: IEEE Signal Processing Magazine.

[35] K. Zhang, Z. Yang, and T. Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," in *Handbook of Reinforcement Learning and Control* (K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever, eds.), pp. 321–384, Cham: Springer International Publishing, 2021.

[36] L. A. Birnbaum, *Machine Learning Proceedings 1993: Proceedings of the Tenth International Conference on Machine Learning, University of Massachusetts, Amherst, June 27-29, 1993*. Morgan Kaufmann, May 2014. Google-Books-ID: TrqjBQAAQBAJ.

[37] A. Torreño, E. Onaindia, A. Komenda, and M. Štolba, "Cooperative Multi-Agent Planning: A Survey," *ACM Comput. Surv.*, vol. 50, pp. 84:1–84:32, Nov. 2017.

[38] Y. Zhou, S. Liu, Y. Qing, K. Chen, T. Zheng, Y. Huang, J. Song, and M. Song, "Is Centralized Training with Decentralized Execution Framework Centralized Enough for MARL?," May 2023. arXiv:2305.17352 [cs].

[39] H.-C. Chen, S.-A. Li, T.-H. Chang, H.-M. Feng, and Y.-C. Chen, "Hybrid Centralized Training and Decentralized Execution Reinforcement Learning in Multi-Agent Path-Finding Simulations," *Applied Sciences*, vol. 14, p. 3960, Jan. 2024. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[40] T. Ikeda and T. Shibuya, "Centralized Training with Decentralized Execution Reinforcement Learning for Cooperative Multi-agent Systems with Communication Delay," in *2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE)*, pp. 135–140, Sept. 2022.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017. arXiv:1707.06347.

[42] S. Iqbal and F. Sha, "Actor-Attention-Critic for Multi-Agent Reinforcement Learning," May 2019. arXiv:1810.02912.

[43] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), pp. 6382–6393, Curran Associates Inc., Dec. 2017.

[44] D. Shi, C. Zhao, Y. Wang, H. Yang, G. Wang, H. Jiang, C. Xue, S. Yang, and Y. Zhang, "Multi actor hierarchical attention critic with RNN-based feature extraction," *Neurocomputing*, vol. 471, pp. 79–93, Jan. 2022.

[45] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," Mar. 2020. arXiv:1706.02275 [cs].

[46] 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," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24611–24624, Dec. 2022.

[47] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark," June 2022. arXiv:2109.14545 [cs].

[48] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017. arXiv:1412.6980 [cs].

[49] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok, "Vmas: A vectorized multi-agent simulator for collective robot learning," *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.

[50] A. Eck, L.-K. Soh, S. Devlin, and D. Kudenko, "Potential-based reward shaping for finite horizon online pomdp planning," *Autonomous Agents and Multi-Agent Systems*, vol. 30, pp. 403–445, 2016.