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

HIPPO-MAT: Decentralized Task Allocation Using GraphSAGE and Multi-Agent Deep Reinforcement Learning

Lavanya Ratnabala[1,*], Robinroy Peter[2,*], Aleksey Fedoseev[1], and Dzmitry Tsetserukou[1]

***Abstract*— We address the problem of decentralized continuous task allocation in heterogeneous multi-agent systems operating in three-dimensional environments. We propose HIPPO-MAT, a fully decentralized framework that combines a GraphSAGE-based graph neural network with Independent Proximal Policy Optimization (IPPO) to enable concurrent and conflict-aware decision-making. Each agent constructs an ego-centric dynamic graph over peers within its communication range, computes embeddings via a mean-aggregating GraphSAGE encoder, and feeds these into its own independent policy. To improve stability, the encoder is pretrained with an encoder–decoder reconstruction loss on synthetic ego-graphs before reinforcement learning using supervised learning. This design allows heterogeneous agents such as unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) to allocate tasks continuously and in parallel without relying on centralized critics or coordination. Navigation is achieved with a reservation-based A* planner coupled with onboard SLAM, ensuring collision avoidance. We validate the approach extensively in simulation with up to 30 agents and in real-world deployment on JetBot ROS AI robots running policies locally on Jetson Nano boards with ESP32-S3 modules for ESP-NOW peer-to-peer communication. Results demonstrate a 91% first-decision conflict-free success rate (CFSR) up to 30 agents and 89.6% on robots, a near-optimal cost gap compared to the centralized Hungarian algorithm, with an average of 16.8%, and significantly faster allocation times.**

Keywords* — *Multi-agent systems, Task Allocation, Deep Reinforcement Learning, Graph Neural Networks, IPPO

I. Introduction

The deployment of heterogeneous multi-agent systems (MAS) composed of unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs) is rapidly increasing in domains such as disaster response, environmental monitoring, and warehouse logistics [1]. Their ability to scale, cooperate [2], operate in a distributed manner, and execute tasks in parallel provides significant efficiency gains [3]. However, allocating tasks continuously and concurrently in dynamic three-dimensional environments remains a formidable challenge. While recent efforts such as DeepFleet [4] validate learning-based fleet coordination at warehouse scale, they rely on centralized models and massive proprietary datasets, limiting applicability in decentralized settings.

In realistic scenarios, new tasks arrive online, multiple agents must make decisions simultaneously, and conflicts can emerge when multiple robots select the same task. Traditional task allocation approaches are either centralized or decentralized. Centralized algorithms, such as the Hungarian method [5], provide optimal global assignments but require global knowledge and high-bandwidth communication, making them brittle and unsuitable for real-time scalability due to single points of failure. Decentralized approaches alleviate these limitations but typically assume sequential processing of tasks or operate in restricted two-dimensional environments. This assumption is impractical in scenarios where tasks are spatially distributed in 3D and require concurrent access. Moreover, many decentralized methods neglect conflict resolution and fail to couple allocation with navigation, leading to collisions and inefficient execution.

To address these shortcomings, we introduce HIPPO-MAT, a decentralized multi-agent deep reinforcement learning (MARL) framework for continuous and parallel task allocation in 3D heterogeneous environments. HIPPO-MAT integrates three key ideas. First, each agent builds its own

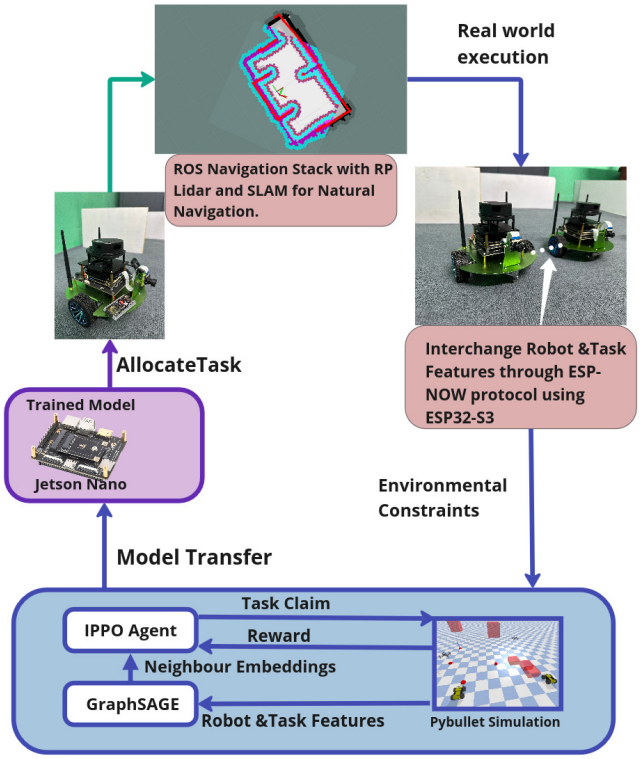

Fig. 1. Experimental setup to test the model in JetBot ROS AI robots equipped with Jetson Nano for individual policy loading and ESP32-S3 for inter-agent communication.

*The authors contributed equally to this work.
[1]The authors are with the Intelligent Space Robotics Laboratory, Skolkovo Institute of Science and Technology, Bolshoy Boulevard 30, bld. 1, 121205, Moscow, Russia.
{lavanya.ratnabala, aleksey.fedoseev, d.tsetserukou}@skoltech.ru
[2]The author is with NeuroFleets (PVT) LTD, Jaffna, Sri Lanka.
robinroy.peter@neurofleets.com

ego-centric dynamic graph that encodes its status, task costs, and task availability relative to its peers. Second, we employ a GraphSAGE encoder with mean aggregation to produce compact embeddings from this ego-graph. The encoder is pretrained with an encoder–decoder reconstruction objective, ensuring stable and expressive representations before MARL training. Third, each agent uses an independent PPO (IPPO) learner trained solely from its local observations and embeddings, avoiding any centralized critic or centralized training with decentralized execution (CTDE). The proposed method has previously demonstrated strong effectiveness in studies on single- and multi-agent drone landing with varying landing pad locations [6]. Building on this, the present work investigates a similar DRL architecture in a more general multi-agent task allocation setting, ensuring complete decentralization in both the learning phase and the deployment phase.

HIPPO-MAT also couples task allocation with motion planning through a reservation-based A* algorithm. Agents dynamically replan in case of conflicts or blockages, ensuring safe navigation while executing tasks. Crucially, we validate HIPPO-MAT not only in large-scale PyBullet simulations with up to 30 heterogeneous agents but also in a real-world deployment using JetBot ROS AI robots.

The main contributions of this work are:

- Continuous and parallel task allocation in 3D: A decentralized MARL framework that enables heterogeneous agents to allocate continuously emerging tasks concurrently in dynamic 3D environments.
- GraphSAGE with independent policy learning: A novel combination of pretrained GraphSAGE mean-aggregating embeddings and IPPO, providing stable and scalable decentralized task allocation.
- Onboard validation: End-to-end real-world deployment on embedded JetBot platforms with SLAM navigation and ESP-NOW communication, demonstrating practical feasibility.

II. Related Works

Multi-Agent Task Allocation (MATA) has been extensively studied in the fields of robotics, autonomous systems, and artificial intelligence. Existing approaches can be broadly categorized into optimization-based methods, market-based mechanisms, and learning-based approaches. Furthermore, distinctions between centralized and decentralized task allocation, as well as homogeneous and heterogeneous multi-agent settings, are critical in determining the efficiency and adaptability of such systems.

Optimization techniques, e.g., the linear sum assignment problem (LSAP) and the Hungarian algorithm, have been widely applied in multi-agent task allocation [7]. While these methods provide optimal solutions, they often rely on centralized computation, rendering them unsuitable for large-scale real-time applications. Ismail et al. [8] proposed a novel decentralized-based Hungarian method to solve this problem. Another decentralized version of the Hungarian method is proposed by Xia et al. [9] to solve task allocation for underwater vehicles. These methods extend the Hungarian approach to decentralized settings by ensuring task allocation remains optimal as long as agent networks remain connected. Kong et al. [10] proposed a new optimization strategy that combines the improved particle swarm optimization and the greedy IPSO-G algorithm. Moreover, two different stochastic approaches, the Genetic Algorithm (GA) and the Ant-Colony Optimization (ACO) algorithm, were introduced by [11] to solve the multi-robot task allocation problem. Peter et al. [12] introduced a swarm intelligence-based task allocation method utilizing decentralized decision-making and subgoal-based path formation.

Market-based strategies, particularly auction-based methods, have been widely explored for decentralized task allocation. Zhong et al. [13] proposed an extended auction-based driver-passenger matching system for ride-sharing applications. Liu et al. [14] extended auction-based methods to multi-UAV systems, incorporating Dubins path-based flight cost estimation to refine task allocations. While auction-based strategies provide high adaptability, their efficiency heavily depends on accurate bid valuation models. An alternative hybrid approach is introduced in the Harmony Drone Task Allocation (DTA) method [15], which integrates a consensus-based auction mechanism with a gossip-based consensus strategy. The Harmony DTA optimizes multi-drone task allocation under complex time constraints by balancing task urgency with resource availability while minimizing communication load.

The integration of DRL into MATA has led to promising advancements, particularly through MARL frameworks. Agrawal et al. [16] introduced an attention-inspired DRL method for warehouse-based task allocation that optimizes decision-making efficiency and scalability. Similarly, Dai et al. [17] propose a decentralized RL approach for heterogeneous multi-robot task allocation, where robots with diverse skills form coalitions to complete tasks, minimizing the overall mission completion time. They use attention mechanisms to reason about task dependencies and agent skills and introduce a constrained flash-forward mechanism to improve training efficiency.

Moreover, recent studies explore Multi-Agent Proximal Policy Optimization (MAPPO) to enhance collaborative decision-making and mitigate non-stationary challenges in multi-agent environments [18]. Lowe et al. [19] apply Q-learning to train cooperative and competitive tasks. However, Christian et al. show that IPPO, where each agent learns its own policy using only local observations, can achieve performance on par with or even better than state-of-the-art centralized training methods (e.g., QMIX, MAPPO) on several challenging maps from the StarCraft Multi-Agent Challenge (SMAC) [20].

Recent advances in graph-based learning methods have demonstrated substantial potential for improving decentralized task allocation. GNNs enable robots to model and process inter-agent relationships, allowing task allocations to be computed based on local graph structures rather than centralized control. Goarin et al. [7] proposed DGNN-GA,

a decentralized GNN-based goal assignment method that optimizes communication efficiency in multi-robot planning for a fixed number of tasks. Heterogeneous multi-agent systems pose additional complexities due to differences in robot capabilities, sensor modalities, and mobility constraints. Blumenkamp et al. [21] developed a real-world framework for deploying decentralized GNN-based policies in multi-robot systems, facilitating seamless sim-to-real transfers. A combination of GNN with ant-colony optimization was explored by Qiu et al. [22].

The GATAR framework [23] further introduced a graph-based task allocation framework for multi-robot target localization, specifically designed for heterogeneous robot systems. Bettini et al. propose Heterogeneous Graph Neural Network Proximal Policy Optimization (HetGPPO), a framework for training heterogeneous MARL policies using Graph Neural Networks (GNNs) for inter-agent communication. By enabling agents to learn distinct behaviors while maintaining fully decentralized training in partially observable environments, HetGPPO highlights the effectiveness of GNNs in decentralized MARL [24]. Ma et al. [25] explored dynamic task allocation for UAVs and UGVs in complex urban environments using an adaptive depth graph neural network (AD-GNN) combined with biomimetic algorithms. Ratnabala et al. [26] recently proposed a GNN-based DRL framework for heterogeneous robot swarms. The suggested MAGNNET approach allowed an efficient compromise between optimization methods and locally optimized greedy approaches; however, the centralized critic model suggests a challenge in further scalability of the system.

Despite significant progress, current approaches exhibit several key limitations. Many continuous task allocation methods assume that tasks are processed sequentially, which is impractical in scenarios where multiple agents must simultaneously access, evaluate, and assign tasks. Additionally, most existing studies [16] assume that only one robot is available at a time and skip the conflict resolution part, failing to capture the complex dynamics of three-dimensional spaces essential for realistic navigation and task allocation. Moreover, task allocation is often decoupled from navigation, even though efficient systems require integrated routing and collision avoidance. Finally, while some methods address heterogeneous teams, they typically rely on centralized information or static task models, limiting scalability in dynamic, large-scale deployments. In contrast, our work proposes a decentralized framework for continuous task allocation in 3D environments by integrating a GraphSAGE-based GNN with an Independent PPO (IPPO) framework.

III. Multi-Agent Task Allocation Approach

A. Problem Formulation

We consider a system of N heterogeneous agents, $\{a_1, a_2, \ldots, a_N\}$, which is operating in a continuous three-dimensional domain $\Omega \subset \mathbb{R}^3$. A stream of M tasks, $\{T_1, T_2, \ldots, T_M\}$, arrives dynamically over time. Each task T_j is characterized by its location $\mathbf{l}_j \in \Omega$ and a binary status $s_j(t) \in \{+1, -1\}$, where $+1$ denotes a *waiting*(unassigned) task and -1 denotes an *assigned* task (already allocated or unavailable). Once a task is allocated, it is replaced by a newly generated task. New tasks are sampled uniformly at random within Ω, independently of agent count or density. Task density is held constant at M = 30 throughout all experiments.

Each agent a_i is described by its position $\mathbf{p}_i(t) \in \Omega$, velocity v_i, and operational status $\sigma_i(t) \in \{-1, +1\}$, where -1 represents the *idle* and $+1$ represents the *assigned* status. Intermediate values are not used, avoiding ambiguity. The travel cost for the agent a_i to complete the task T_j is defined as:

$$c_{ij}(t) = \frac{d_{ij}(t)}{v_i}, \tag{1}$$

where $d_{ij}(t)$ is the shortest-path distance (computed via A*). Costs are clipped to $[0, c_{\max}]$ and normalized to $[-1, 1]$ for training stability.

Rather than solving a centralized assignment with binary variables, each agent independently selects an action from:

$$A_i = \{0, 1, \ldots, M\}, \tag{2}$$

where $a_i(t) = 0$ means "no task requested," and $a_i(t) = j$ means "requesting task T_j". If multiple agents request the same task, the allocation is resolved by cost comparison, and the lower-cost agent succeeds; others incur a penalty.

B. Ego-Centric Graph Construction

At each step, every agent a_i builds its own ego-centric dynamic graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$. Nodes correspond to agents only. At each control step, the ego agent forms edges to all peers currently within its communication radius:

$$\mathcal{E}_i = \{(v_i, v_j) \,|\, j \in \mathcal{N}(i)\},$$

where $\mathcal{N}(i)$ denotes the set of peers within communication range of a_i at that step.

No central graph server is required, and ego-graphs are rebuilt at every control step. We do not use fixed-k selection or random neighbor subsampling; instead, $\mathcal{N}(i)$ contains precisely the peers within communication range of a_i at that time, so the neighborhood size varies with team configuration and positions. During training, the communication radius was randomized across episodes for domain randomization, exposing the policy to varying neighborhood sizes. At evaluation, a fixed radius was used for all experiments. Since the mean-aggregating GraphSAGE encoder is permutation-invariant and supports variable-degree graphs, the learned policy generalizes across different communication ranges without retraining. A lightweight warehouse management system (WMS) validates assignments in a first-in, first-out (FIFO) order: the first claim succeeds, and simultaneous claims are rejected. Agents decide via HIPPO-MAT but confirm with the WMS to avoid duplicate allocations.

The raw observation vector of the agent a_k is

$$\mathbf{x}_k \in \mathbb{R}^{1+2M} = \big[\sigma_k, \tilde{c}_{k1}, \ldots, \tilde{c}_{kM}, \chi_1, \ldots, \chi_M\big], \tag{3}$$

where σ_k is the agent's binary status, $\tilde{c}_{kj}$ are the normalized travel costs to tasks, and $\chi_j \in \{-1, +1\}$ is the task activity.

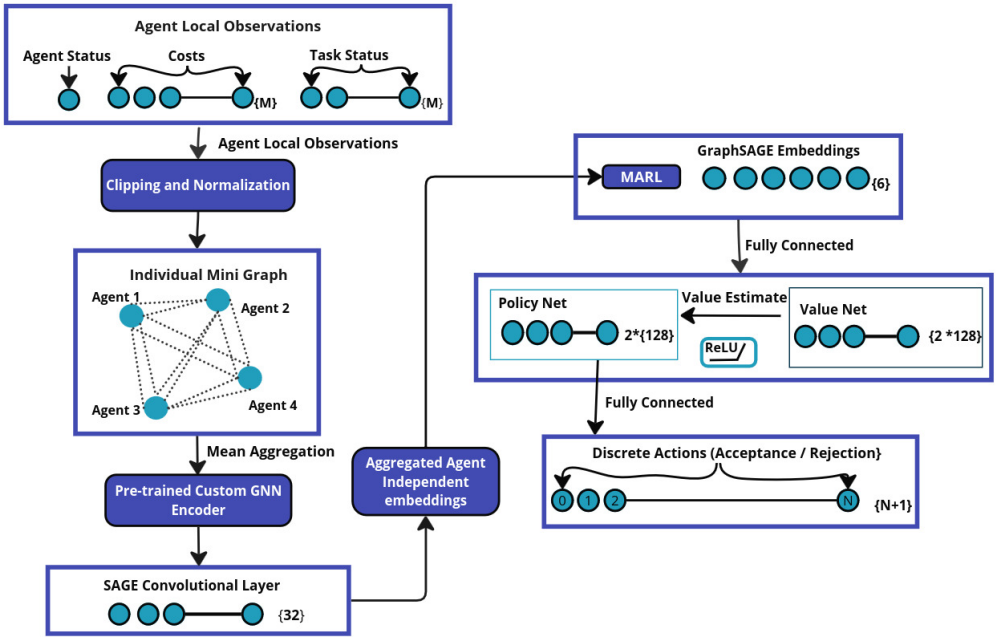

Fig. 2. Multi-agent reinforcement learning and GraphSAGE architecture for task allocation.

C. Graph Neural Network Encoding

We employ GraphSAGE with mean aggregation to encode ego-graphs. The embedding for the agent a_i is:

$$\mathbf{z}_i = \tanh\left(\mathbf{W}\mathbf{x}_i + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\mathbf{W}'\mathbf{x}_j\right), \tag{4}$$

where $\mathcal{N}(i)$ are all neighbors of i, $\mathbf{W}, \mathbf{W}' \in \mathbb{R}^{d_z\times(2M+1)}$ are trainable matrices, and $d_z = 6$ defines the latent size. Thus $\mathbf{z}_i \in \mathbb{R}^6$ becomes a compact state embedding. We set $d_z = 6$ after a sweep over $\{4, 6, 8, 12\}$ on validation runs (cost and CFSR); $d_z = 6$ balanced accuracy and latency and was kept fixed for all experiments.

Unlike the original GraphSAGE setting that often uses random neighbor subsampling, we do not subsample: mean aggregation is computed over all communication-range neighbors in $\mathcal{N}(i)$ present at that step.

We used GraphSAGE because each agent builds its own ego-centric graph and generates a fixed-size embedding that encodes both its own state and the states of its neighbors. This embedding serves as a token, enabling the agent to learn policies that remain consistent across variable team sizes while supporting fully decentralized task allocation.

1) Pretraining: To stabilize training, we pretrained the encoder with an autoencoder objective on synthetic ego-graphs. The encoder produces $\mathbf{z}_i$, and a decoder reconstructs $\hat{\mathbf{x}}_i$, minimizing $\sum_i \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$. Pretrained weights initialize $\mathbf{W}, \mathbf{W}'$, ensuring meaningful embeddings before reinforcement learning begins. We use a single-layer GraphSAGE encoder–decoder (PyG `SAGEConv`) with input dimension $2M + 1$ and hidden size 50, optimized with Adam (lr $= 10^{-3}$), batch size 32, for 50 epochs, and additive Gaussian noise $\mathcal{N}(0, 0.1^2)$ on inputs; synthetic ego-graphs are generated per mini-batch with a complete (fully connected) edge index, and checkpoints are saved and later loaded to initialize HIPPO-MAT. The mean-aggregating GraphSAGE encoder is permutation-invariant and naturally supports variable-degree graphs, so no padding or truncation is needed when $|\mathcal{N}(i)|$ changes.

D. Policy Learning with IPPO

Each agent is equipped with an independent policy π_i and value function V_i, trained via Independent Proximal Policy Optimization (IPPO). The observation is:

$$\mathbf{o}_i(t) = \big[\mathbf{x}_i(t), \mathbf{z}_i(t)\big], \tag{5}$$

and the agent samples action $a_i(t) \in A_i$ from π_i.

1) Reward Function: The reward $r_i(t)$ is defined as:

$$r_i(t) = \begin{cases} -c_{ij}(t), & \text{if } a_i \text{ is allocated task } T_j, \\ -\lambda, & \text{if multiple agents request } T_j, \\ -\mu, & \text{if idle while tasks are available,} \\ +\eta, & \text{if valid allocation is low-cost.} \end{cases} \tag{6}$$

Here, μ penalizes unjustified idling. Let $\mathcal{W} = \{j \mid \chi_j = +1\}$ be the set of waiting tasks, $c_i^{\min} = \min_{j\in\mathcal{W}} c_{ij}$ the best available cost for the idle agent i, and $c^{\min} = \min_{k:\,\sigma_k=-1} c_k^{\min}$ the best across idle agents. If $a_i(t) = 0$ while $c_i^{\min} \le c^{\min}+\varepsilon$ (we use a small tolerance ε, e.g., $0.05\,c_{\max}$), we apply the penalty $-\mu$; otherwise, idling is not penalized. In all experiments we set $\lambda = 200$, $\mu = 100$, $\eta = 100$, and $c_{\max} = 15.0$. A global team reward $R(t) = \sum_i r_i(t)$ is logged for training analysis.

2) Policy Architecture and Training: The policy and value networks are two-layer MLPs with 128 ReLU neurons per layer. The policy outputs a softmax distribution over A_i, and the value network outputs a scalar $V_i(t)$. Training uses Adam with a learning rate 10^{-5}, the rollout length 100, the batch size 1000, and the entropy coefficient 0.05. Generalized Advantage Estimation (GAE, $\lambda = 0.95$) and the discount factor $\gamma = 0.99$ stabilize learning. After training, agents act independently using only their local observations. No centralized critic or coordinator is required, guaranteeing scalability and robustness to communication loss. The overall neural network architecture is shown in Fig. 2.

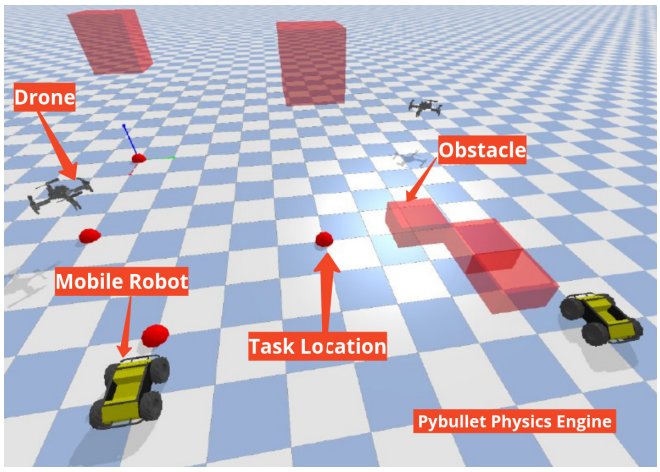

Fig. 3. Simulation environment in PyBullet. Agents navigate with dynamic task assignments and obstacles.

Although IPPO does not explicitly address the non-stationarity introduced by concurrent learning agents, three properties of HIPPO-MAT mitigate this concern. First, the ego-centric GraphSAGE embedding (Eq. 4) aggregates each neighbor's status and cost features, implicitly conditioning every agent's observation on its peers' current states; this reduces apparent non-stationarity by making transitions more predictable from each agent's viewpoint. Second, pretraining the encoder with the reconstruction objective stabilizes the embedding space before MARL training begins, shortening the initial phase during which all policies change rapidly. Third, consistent with de Witt et al. [20], who demonstrated that IPPO matches or outperforms centralized training methods such as QMIX and MAPPO on cooperative benchmarks, attributing this to PPO's policy clipping providing inherent robustness to non-stationarity, our results confirm that independent learning with enriched local observations via GNN embeddings is sufficient for stable decentralized training in this task-allocation domain. The reinforcement learning pipeline in this work extends the single-agent DRL architecture of [6], [27] to the multi-agent setting with independent policies and graph-based embeddings.

E. Navigation and Re-planning

Once HIPPO-MAT assigns the task T_j to the agent a_i, the assignment message immediately triggers the onboard reservation-based A* planner. The planner computes a time-resolved shortest path from $\mathbf{p}_i(t)$ to $\mathbf{l}_j$ while inserting the sequence of grid cells into a shared reservation table indexed by (x, y, z, t). If another agent already occupies a cell at the intended arrival time, a conflict flag is raised; the blocked agent invokes local re-planning with a 200 ms timeout. After three consecutive blockings, we inflate the traversal cost of the offending region by 3× and rerun A*, forcing the agent to select an alternative corridor. This tight allocate-then-plan loop ensures that task-allocation decisions are always grounded in reachable, collision-free motion and distinguishes transient congestion from persistent deadlocks.

IV. Experiments

A. Experimental Setup

We evaluated HIPPO-MAT in both simulation and real-world settings. Simulation experiments were conducted in a $50 \times 50 \times 30$ m 3D PyBullet environment with a mix of ground robots ($v \approx 3$ m/s) and aerial drones ($v \approx 5$ m/s). At any given time, the system maintained 30 active tasks; as soon as a task was allocated, a new one was generated. Fig. 3 shows our simulation environment. Random initialization of agent positions, task locations, and obstacles is performed at the start of each episode to ensure robustness.

Efficient navigation was handled with a reservation-based A* planner. Replanning was triggered upon conflicts. To validate real-world transfer, we deployed the trained policies directly on JetBot ROS AI Robots, each equipped with a Jetson Nano for onboard inference and an ESP32-S3 for communication using the ESP-NOW protocol. Fig. 4 shows the real setup, while Fig. 5 shows an example SLAM-based environment map.

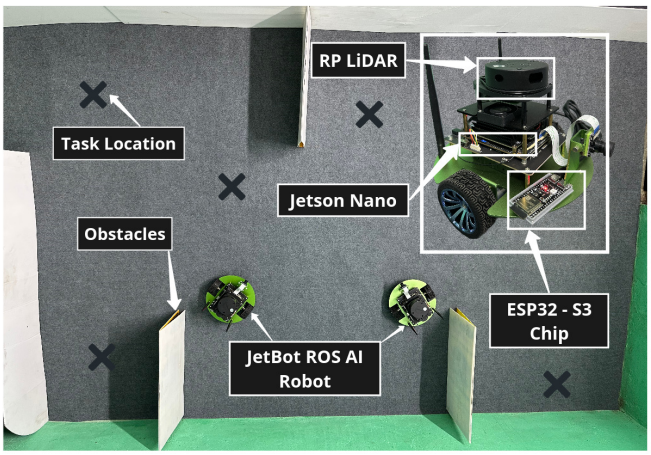

Fig. 4. JetBot ROS AI robots equipped with Jetson Nano for individual policy loading and ESP32-S3 for inter-agent communication.

We evaluated three primary metrics:

- Total travel cost (Cost): the sum of all agent travel times.
- Conflict-free success rate (CFSR): an instantaneous, one-shot metric computed from the very first action output of the policy at the allocation step.

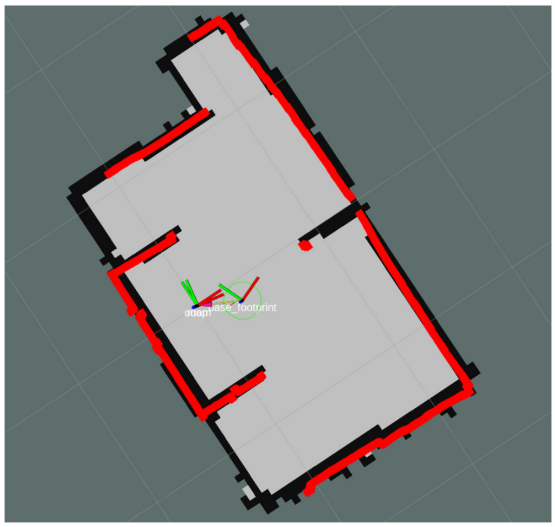

Fig. 5. Environment mapping by JetBot ROS AI robot.

- Responsiveness: in fixed task allocation we report allocation time (wall-clock time to allocate the entire pool); in continuous task allocation we report mean decision latency per task, defined as the average time from task appearance to the policy's first action for that task.

Metric semantics: Unless otherwise stated, the CFSR reported in Tables III and IV is the first-decision rate measured at $t{=}0$, meaning the first policy output. In subsequent control cycles, residual conflicts are resolved, and the final allocation becomes conflict-free in all tested runs. We emphasize the first-decision rate because fast, low-latency dispatch is the key requirement in time-critical applications such as warehouse robotics or UAV swarms. For each agent count, we report the mean CFSR over 10 runs. In fixed, each run has N tasks (equal to the number of robots). In continuous, each run processes 100 tasks; CFSR is the fraction assigned without first-decision conflicts.

We compare HIPPO-MAT against classical, heuristic, and learning-based approaches:

- Hungarian Algorithm: centralized optimal solution to the linear sum assignment problem.
- IPSO-G [10]: a heuristic strategy combining particle swarm optimization with greedy task allocation.
- Random Assignment: tasks are allocated uniformly at random.
- MAGNNET [26]: centralized training with a decentralized execution method used for task allocation with MAPPO and GCN.

B. Simulation Results

1) Training Dynamics: Fig. 6 shows the mean reward increasing over training steps, while entropy decreases (Fig. 7), indicating a transition from exploratory to confident policies.

2) Fixed Task Allocation: Table I reports total travel costs. HIPPO-MAT approaches Hungarian performance and consistently outperforms heuristic baselines and Random.

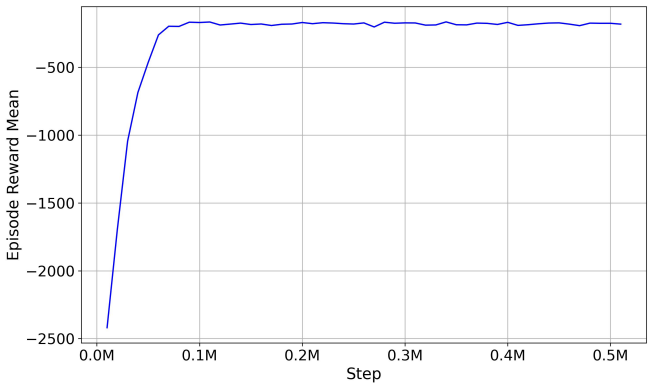

Fig. 6. Mean reward vs. training steps during learning.

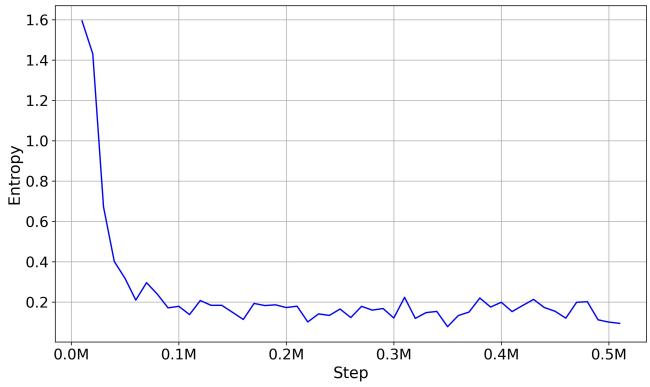

Fig. 7. Entropy vs. training steps during learning.

TABLE I: TOTAL TRAVEL COST IN FIXED TASK ALLOCATION VS. NUMBER OF AGENTS.

N Agents / Method	5	10	20	30
Hungarian	25.0	62.0	140.0	260.0
HIPPO-MAT	25.3	64.5	153.0	312.0
MAGNNET	26.1	68.0	161.5	326.0
IPSO-G	27.0	70.5	168.5	348.0
Random	31.8	78.0	182.0	402.0

3) Continuous Task Allocation: Table II summarizes results for continuously replenished tasks. HIPPO-MAT maintains strong performance and scales well up to 30 agents, outperforming MAGNNET, IPSO-G, and Random Assignment. In addition to cost, we measured mean decision latency per task. HIPPO-MAT consistently achieved sub-second latency (0.2–0.5 s) even with 30 agents, whereas MAGNNET required longer (0.6–1.5 s) due to centralized critics or weaker feature encoders. This responsiveness highlights HIPPO-MAT's ability to operate in online continuous allocation, where tasks arrive in real time.

4) Conflict-Free Success Rate and Decision Latency (Continuous): Tables III and IV present success rates and responsiveness. HIPPO-MAT sustains $\sim$ 90% first-decision conflict-free allocations even at scale. Hungarian achieves 100% by construction. Percentages are averages across 10 runs (with N tasks per run for fixed and 100 tasks per run

TABLE II: TOTAL TRAVEL COST IN CONTINUOUS TASK ALLOCATION VS. NUMBER OF AGENTS.

Method \ N Agents	5	10	20	30
Hungarian	281.6	560.3	1931.2	4524.3
HIPPO-MAT	295.6	612.1	2236.2	6212.4
MAGNNET	302.4	642.1	2382.6	6802.1
IPSO-G	308.3	667.2	2445.0	7201.1
Random	484.4	736.5	3282.3	9453.2

for continuous), so they reflect integer ratios of successfully assigned tasks. For the continuous setting, we additionally report mean decision latency per task (lower is better); HIPPO-MAT remains below 0.5 s up to 30 agents.

TABLE III: CONFLICT-FREE SUCCESS RATE AND ALLOCATION TIME FOR FIXED TASK ALLOCATION VS. NUMBER OF AGENTS.

Metric	Method	5	10	20	30
Success Rate (%)	Hungarian	100	100	100	100
	HIPPO-MAT	98	96	90	82
	MAGNNET	94	86	75	68
	IPSO-G	90	82	74	62
	Random	62	55	44	32
Allocation Time (s)	Hungarian	0.8	1.5	2.8	5.6
	HIPPO-MAT	0.2	0.2	0.4	0.5
	MAGNNET	0.5	0.7	1.6	3.2
	IPSO-G	0.3	0.4	0.7	1.2
	Random	0.1	0.2	0.4	0.9

TABLE IV: CONFLICT-FREE SUCCESS RATE (FIRST DECISION) AND MEAN DECISION LATENCY FOR CONTINUOUS TASK ALLOCATION VS. NUMBER OF AGENTS.

Metric	Method	5	10	20	30
Success Rate (%)	Hungarian	100	100	100	100
	HIPPO-MAT	98	94	89	81
	MAGNNET	90	82	71	64
	IPSO-G	84	76	66	55
	Random	52	44	36	28
Decision Latency (s)	Hungarian	0.7	1.3	2.5	5.1
	HIPPO-MAT	0.2	0.3	0.4	0.5
	MAGNNET	0.5	0.7	1.6	3.1
	IPSO-G	0.3	0.5	0.8	1.3
	Random	0.2	0.3	0.6	1.0

C. Real-World Validation

The trained HIPPO-MAT model was deployed without modification on JetBot ROS AI Robots. Each robot ran its policy locally on a Jetson Nano and communicated peer-to-peer over ESP-NOW. Tasks were randomly generated in an indoor environment and mapped online via SLAM (Fig. 5). HIPPO-MAT successfully handled sensor noise, partial observability, and communication delays, sustaining a first-decision CFSR of 89.6%, closely matching simulation. With a few subsequent policy cycles, all conflicts were resolved, yielding fully conflict-free allocations. Decision latency remained within 0.4 s on average, validating that the onboard inference pipeline meets real-time constraints.

D. Discussion

HIPPO-MAT balances performance and scalability: while the centralized Hungarian method remains globally optimal, HIPPO-MAT achieves near-optimal allocations with far lower decision latency. Compared to MAGNNET, which requires centralized critics and global information, our fully independent GraphSAGE+IPPO design is lighter and scales linearly. HIPPO-MAT minimizes not only travel costs but also decision latency. In continuous allocation, where allocation time is undefined, HIPPO-MAT achieves average decision latencies of 0.2–0.5 s depending on team size. This ensures that tasks are dispatched in near real time, which is essential for safety-critical domains such as warehouse robotics or drone swarms.

The comparison with MAGNNET [26], which uses GCN with a centralized MAPPO critic, serves as an indirect architectural ablation. HIPPO-MAT consistently outperforms MAGNNET across all metrics, confirming that GraphSAGE with independent policy learning is more effective for continuous decentralized task allocation. A full component-wise ablation is planned as future work.

Beyond 30 agents, more peers fall within the fixed communication radius, producing denser ego-centric graphs where mean aggregation dilutes individual agent signals, leading to higher conflict rates. This limitation stems from neighborhood density rather than agent count itself; in a larger spatial domain with the same density, performance would be expected to remain stable. We identified attention-based or adaptive aggregation as future work to further strengthen scalability. Because HIPPO-MAT is fully decentralized, it cannot guarantee globally optimal cost; in practice its travel cost is higher than the centralized Hungarian method, with the gap widening as team size or task churn increases (see Tables I and II).

V. CONCLUSION AND FUTURE WORK

This paper introduces HIPPO-MAT, a decentralized framework for continuous task allocation in heterogeneous multi-agent systems. By combining per-agent dynamic graphs with mean-aggregated GraphSAGE embeddings and independent PPO, our approach enables UAVs and UGVs to concurrently allocate tasks in dynamic 3D environments without centralized control. A reservation-based A* planner provides efficient routing and collision avoidance. Extensive experiments highlight that HIPPO-MAT achieves a near-optimal cost gap relative to the centralized Hungarian method: on average 9% in fixed task scenarios and 17% in continuous task allocation while clearly outperforming heuristic and learning baselines. In terms of allocation quality, HIPPO-MAT sustains high first-decision conflict-free success rates of around 91%, with all residual conflicts resolved in subsequent cycles. Moreover, the framework delivers strong responsiveness, maintaining sub-second decision latency per task (0.2–0.5 s) even with 30 agents. Real-world validation on JetBot ROS AI Robots (Jetson Nano + ESP32-S3/ESP-NOW) confirms the practicality of deploying HIPPO-MAT under realistic sensing and communication constraints.

Future work will explore replacing mean aggregation with adaptive or attention-based mechanisms to enhance scalability beyond 40 agents, incorporating richer heterogeneous features such as task priorities and energy constraints, and extending real-world trials to larger robot fleets with advanced SLAM and multi-hop communication.

ACKNOWLEDGEMENTS

Research reported in this publication was financially supported by the RSF grant No. 24-41-02039.

REFERENCES

[1] C. Cai, G. Lu, C. Liu, W. Wang, Y. Li, and H. Song, "Multi-agent heterogeneous hybrid safe logistics sorting system based on transformers," in *Proc. Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2025, pp. 1–6.

[2] A. Gupta, E. Dorzhieva, A. Baza, M. Alper, A. Fedoseev, and D. Tsetserukou, "Swarmhawk: Self-sustaining multi-agent system for landing on a moving platform through an agent supervision," in *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 990–997.

[3] R. Zakir, M. Salahshour, M. Dorigo, and A. Reina, "Heterogeneity can enhance the adaptivity of robot swarms to dynamic environments," in *Proc. Int. Conf. on Swarm Intelligence (ANTS)*, 2024, p. 112–126.

[4] A. Agaskar, S. Siva, W. Pickering, K. O'Brien, C. Kekeh, A. Li *et al.*, "Deepfleet: Multi-agent foundation models for mobile robots," 2025, arXiv:2508.08574.

[5] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955.

[6] R. Peter, L. Ratnabala, D. Aschu, A. Fedoseev, and D. Tsetserukou, "Lander.ai: Drl-based autonomous drone landing on moving 3d surface in the presence of aerodynamic disturbances," in *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2024, pp. 295–300.

[7] M. Goarin and G. Loianno, "Graph neural network for decentralized multi-robot goal assignment," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4051–4058, May 2024.

[8] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *Proc. Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, Jun. 2017, pp. 23–28.

[9] M. Xia, "Research on decentralized task allocation and collaboration based on multiple auvs," in *Proc. Int. Signal Processing, Communications and Engineering Management Conference (ISPCEM)*, Montreal, QC, Canada, 2023, pp. 274–281.

[10] X. Kong, Y. Gao, T. Wang, J. Liu, and W. Xu, "Multi-robot task allocation strategy based on particle swarm optimization and greedy algorithm," in *Proc. IEEE Joint Int. Information Technology and Artificial Intelligence Conf. (ITAIC)*, 2019, pp. 1643–1646.

[11] M. Shelkamy, C. M. Elias, D. M. Mahfouz, and O. M. Shehata, "Comparative analysis of various optimization techniques for solving multi-robot task allocation problem," in *Proc. Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 538–543.

[12] R. Peter, L. Ratnabala, E. Y. Andrew Charles, and D. Tsetserukou, "Dynamic subgoal based path formation and task allocation: A neurofleets approach to scalable swarm robotics," in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC)*, Kuching, Malaysia, 2024, pp. 878–883.

[13] Y. Zhong, L. Gao, T. Wang, S. Gong, B. Zou, and D. Yu, "Achieving stable and optimal passenger-driver matching in ride-sharing system," in *Proc. IEEE Int. Conf. on Mobile Ad Hoc and Sensor Systems (MASS)*, Oct. 2018, pp. 125–133.

[14] J. Liu, Z. Zhang, T. Xu, and C. Yan, "Multi-uav united task allocation via extended market mechanism based on flight path cost," in *Proc. IEEE Int. Conf. on Unmanned Systems (ICUS)*, Oct. 2024, pp. 50–55.

[15] M. Eser and A. E. Yilmaz, "A gossip-based auction algorithm for decentralized task rescheduling in heterogeneous drone swarms," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–25, 2025.

[16] A. Agrawal, A. S. Bedi, and D. Manocha, "Rtaw: An attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, Jun. 2023, pp. 1393–1399.

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

[18] 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 *Advances in Neural Information Processing Systems*, Nov. 2022, vol. 35, pp. 24 611–24 624.

[19] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Int. Conf. on Neural Information Processing Systems (NIPS)*, 2017, p. 6382–6393.

[20] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, "Is independent learning all you need in the StarCraft multi-agent challenge?" 2020, arXiv:2011.09533.

[21] J. Blumenkamp, S. Morad, J. Gielis, Q. Li, and A. Prorok, "A framework for real-world multi-robot systems running decentralized gnn-based policies," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, May 2022, pp. 8772–8778.

[22] J. Qiu, Y. Liu, Y. Yu, and W. Li, "Graph convolutional network based ant colony optimization for robot task allocation," in *Proc. IEEE Symposium Series on Computational Intelligence (SSCI)*, 2023, pp. 985–991.

[23] J. Peng, H. Viswanath, and A. Bera, "Graph-based decentralized task allocation for multi-robot target localization," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 10 676–10 683, Nov. 2024.

[24] M. Bettini, A. Shankar, and A. Prorok, "Heterogeneous multi-robot reinforcement learning," in *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2023, p. 1485–1494.

[25] Z. Ma, J. Xiong, H. Gong, and X. Wang, "Adaptive depth graph neural network-based dynamic task allocation for uav-ugvs under complex environments," *IEEE Transactions on Intelligent Vehicles*, vol. 10, no. 5, pp. 3573–3586, 2025.

[26] L. Ratnabala, A. Fedoseev, R. Peter, and D. Tsetserukou, "Magnnet: Multi-agent graph neural network-based efficient task allocation for autonomous vehicles with deep reinforcement learning," in *Proc IEEE Intelligent Vehicles Symposium (IV)*, Apr. 2025, pp. 970–975.

[27] R. Peter, L. Ratnabala, D. Aschu, A. Fedoseev, and D. Tsetserukou, "Tornadodrone: Bio-inspired drl-based drone landing on 6d platform with wind force disturbances," in *Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, 2024, pp. 516–521.