What Can They Do? Task Planning using Affordances in Heterogeneous Multi-Robot Teams

Tabitha Oanda
Department of Computer Science
Brown University, United States
tabitha_oanda@brown.edu

Abstract: We present an affordance-augmented approach to heterogeneous multi-robot task allocation (MRTA) within a decentralized partially observable MDP framework. Our key contribution is an affordance computation algorithm that enables robots to implicitly model their teammates' behaviors, producing a probability distribution over targets based on capability matching and spatial relationships. By integrating these affordance values into observations, robots can make more informed allocation decisions without explicit communication. Implementing this approach with Multi-Agent Proximal Policy Optimization (MAPPO), we demonstrate that affordance-augmented observations maintain task completion performance while achieving approximately twice the sample efficiency during training compared to standard observations. Our results suggest that affordance augmentation effectively guides policy learning toward promising coordination strategies by providing implicit teammate modeling capabilities to individual robots in heterogeneous teams.

1 Introduction

Coordination plays a vital role in efficient multi-robot task allocation (MRTA). In these settings, robot teams must execute tasks independently while avoiding collisions and maintaining resilience against routine interferences. This capability is crucial in environments like hospitals, where medical personnel and robots work together to transport supplies, or in construction sites, where diverse robots conduct inspections across different structures. However, these domains face significant challenges from unpredictable interferences that can severely impact team performance.

We address MRTA through the lens of decentralized partially observable MDPs (Dec-POMDPs), focusing on heterogeneous multi-robot teams that must visit targets with varying capability requirements. In our framework, agents can observe each other's positions but don't know their teammates' intentions or planned paths. To enhance coordination without explicit communication, we propose an affordance computation algorithm that enables robots to model their teammates' likely behaviors. This algorithm outputs a probability distribution over potential targets based on capability matching and spatial positioning, creating a richer observation space that implicitly encodes team dynamics (Section 4).

The flexibility of our decentralized policy allows any agent to adopt it during execution and coordinate with others, regardless of the policies other agents might be using. Our approach eliminates direct agent communication, instead relying on implicit signaling through observations, similar to how new team members learn by watching and adapting to experienced colleagues. While agents cannot control their teammates directly, their actions can influence teammates' behaviors, highlighting the importance of effective coordination despite the absence of explicit communication.

Our experiments evaluate the affordance computation algorithm both as an observation enrichment mechanism for reinforcement learning and as a standalone heuristic algorithm (Section 5). We implement this approach using Multi-Agent Proximal Policy Optimization (MAPPO), comparing

affordance-augmented observations against standard observations. Results demonstrate that affordance augmentation maintains comparable task completion performance while achieving significantly faster convergence—approximately twice as sample efficient as the non-augmented baseline (Section 6). This suggests that our affordance algorithm effectively guides the learning process toward promising regions of the policy space, enabling more efficient acquisition of effective coordination strategies for multi-robot teams. The project code is found is this *private* Github repository.

2 Related Work

Heuristic policies are feasible for small and highly predictable environments. In the case of several dynamic agents, targets, and highly stochastic environments, it is useful to use neural networks to approximate policies for task allocation that meet specific constraints such as minimizing makespan, maximizing reward or agent utility, minimizing collisions, etc. Unlike Liu et al. [1], whose agents use a greedy algorithm for cooperative action selection within a shared-reward framework, our approach incorporates contextual affordances to improve decision quality. While Agrawal et al. [2] learn task allocation policies based on feedback from predetermined navigation algorithms, our method develops more adaptive behaviors through affordance computation. Similarly, where Amini et al. [3] apply Partially Observable Monte Carlo Planning with greedy estimation of teammate actions in grid-world environments, our approach refines the greedy algorithm with an anticrowding affordance algorithm, enabling agents to prioritize tasks they can complete more efficiently than teammates.

Visual-Language Models (VLM) and Large Language Models (LLM) have been instrumental in expanding the capabilities of agent reasoning for task decomposition and task allocation. Ahn et al. [4] proposed a distributed task and motion planning (TAMP) method in which agents used a VLM to process their observations within their field-of-view to plan for task selection and navigation. Kannan et al. [5] developed a centralized LLM planning system for task decomposition and coalition formation, sacrificing the decentralized resilience our method provides. Thomas et al.[6] use a VLM to extract scene objects before LLM-based task generation, requiring multiple model queries for sophisticated semantic reasoning, whereas our method trades some of this advanced reasoning for significantly faster execution by using a deterministic affordance algorithm with pre-classified objects. Similarly, Liu et al. [7] employ LLMs to compute 'agent-to-object' compatibility scores for heterogeneous teams in tidy-up tasks, introducing latency issues, whereas our method employs a computationally efficient affordance algorithm that operates with low overhead, prioritizing real-time performance over the nuanced object-task semantic understanding that LLMs provide.

Our task allocation approach is modeled as a task and motion planning (TAMP) problem, drawing inspiration from multi-agent path finding (MAPF) research [8, 9, 10]. We implement a variation of Proximal Policy Optimization (PPO) extended for multiple agents to learn a shared policy—building on Skrynnik et al. [10], who fragment the MAPF problem into two distinct components: using traditional heuristic planning (e.g., A-star) to generate plans for sub-goals, and employing reinforcement learning for collision avoidance. To evaluate our multi-agent reinforcement learning (MARL) approach against heuristic planning alternatives, we adopt the metrics proposed by Lau et al. [11]. While most existing methods assume closed teams and/or inter-agent communication for task assignments, our work distinctly focuses on teammate modeling to enable dynamic team formation and scaling without explicit communication, addressing a crucial gap in decentralized multi-robot coordination systems.

3 Problem Definition

3.1 Preliminaries: Markov Decision Process Formulation

We begin with a standard Markov Decision Process (MDP) formulation referenced in Littman et al. [12]. An MDP is defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ denotes the state space and $\mathcal{A}$ represents the action space. The transition function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, gives the

probability of transitioning to state s' when executing action a from state s. The reward function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, provides a scalar reward for taking action a in state s and arriving at state s'. Finally, $\gamma \in [0, 1]$ is the discount factor for future rewards. In this framework, an agent's goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maps states to actions and maximizes the expected discounted return $\mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$, where the expectation is taken over trajectories τ generated by following policy π. To maximize the expected discounted return, the agents rely on a value function that estimates this return as $V^\pi(s) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t, s_{t+1})|s_0 = s]$.

In many practical scenarios, agents cannot directly access the complete state information, leading to partial observability. The multi-agent adaptation of this limited information setting can be structured as a decentralized partially observable MDP (DEC-POMDP) [13, 14]. Our DEC-POMDP formulation consists of the tuple $\langle S, \{A_i\}_{i=1}^n, \{O_i\}_{i=1}^n, P, R, \gamma\rangle$. The global environment is represented as the state space S, while each agent i has an individual action space A_i and observation space O_i. An observation function maps the global state to each agent's limited view: agent i receives observation o_i based on the current state. State transitions follow $P(s'|s, \mathbf{a})$, which determines the probability of moving from state s to s' when agents collectively take joint action $\mathbf{a} = (a_1, a_2, ..., a_n)$. Agents individually select actions according to their policies $\pi_i : O_i \to A_i$ with the shared objective of maximizing the expected discounted return $\mathbb{E}_{\tau\sim\{\pi_i\}_{i=1}^n}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{A}_t, s_{t+1})]$, where $\mathbf{A}_t = (a_1^t, a_2^t, ..., a_n^t)$ is the joint action at time t and the expectation is over trajectories generated by the joint policy.

3.2 Problem Formalization

Our problem formulation assumes agents operate without knowledge of their teammates' goals and have no direct control over other agents' actions. The agents are solving a spatial target visitation problem where each target has a set of requirements represented by scalar values. The team consists of heterogeneous agents, each with distinct capabilities, also represented by scalar values. The global task is complete when all targets have been visited and their requirements satisfied by the visiting agents. The primary objective is to minimize the total time Z taken to complete all tasks, which implicitly maximizes utility and minimizes idle time.

We formalize this problem mathematically as follows: Let x_j^i be a binary variable where $x_j^i = 1$ if robot i completes target j. We define R as the set of all robots, T as the set of all targets, and K as the set of robot capability types. For each robot $i \in R$, its capability of type $k \in K$ is denoted by b_i^k. Similarly, for each target $j \in T$, the required capability of type $k \in K$ is denoted by a_j^k, and c_j^i represents the time for robot i to complete target j.

A robot must meet or exceed all task requirements to be assigned a target, such that:

$$b_i^k \geq a_j^k \cdot x_j^i \; \forall i \in R, j \in T, k \in K \tag{1}$$

Each task is completed by exactly one robot, such that:

$$\sum_{i\in R} x_j^i = 1 \; \forall j \in T \tag{2}$$

The primary objective is to minimize the makespan (the completion time of the last agent):

$$\min Z, \text{ where } Z \geq \sum_{j\in T} c_j^i \cdot x_j^i \; \forall i \in R \tag{3}$$

The secondary objective is to maximize the number of completed targets:

$$\max \sum_{i\in R} \sum_{j\in T} x_j^i \tag{4}$$

This formulation captures the key aspects of our multi-robot task allocation problem, where agents with heterogeneous capabilities must collectively visit and satisfy the requirements of multiple targets. The DEC-POMDP framework provides the theoretical foundation for learning effective policies in this partially observable multi-agent setting.

3.3 Multi-Agent Proximal Policy Optimization (MAPPO)

We select MAPPO as our neural approximator based on empirical evidence from Yu et al. [15] demonstrating its superior performance over DQN networks in multi-particle environments. To solve the DECPOMDP presented in subsection 3.1, MAPPO uses a "centralized training with decentralized execution" paradigm where agents employ policies $\pi(a_i|o_i)$ parameterized by θ to produce primitive navigation actions from local observations. The objective is to maximize the discounted accumulated reward $J(\theta) = \mathbb{E}[\sum_t \gamma^t R(s_t, A_t)]$, where $A_t = (a_t^1, \ldots, a_t^n)$ represents the joint action at time step t.

This approach directly connects to our MRTA problem formulation in subsection 3.2. The state space S incorporates information about robot positions, capabilities b_i^k, and target requirements a_j^k from our formal model. The action space consists of primitive navigation actions that agents use to maneuver toward and complete tasks matching their capabilities. The reward function incorporates both our makespan objective (Equation 3) and capability constraints (Equation 2). MAPPO's architecture learns both an actor network parameterized by θ that maps observations to action probabilities and a critic network parameterized by ϕ that evaluates action sequences over planning horizons.

MAPPO's actor network is trained to maximize $L(\theta) = \mathbb{E}[\min(r_{\theta,i} \cdot A_i, \text{clip}(r_{\theta,i}, 1-\varepsilon, 1+\varepsilon) \cdot A_i)] + \sigma \cdot \mathbb{E}[S[\pi_\theta(o_i)]]$, where the importance sampling ratio $r_{\theta,i}$ and advantage function A_i guide policy improvement toward our optimization objectives. Concurrently, the critic network minimizes $L(\phi) = \mathbb{E}[\max[(V_\phi(s_i) - \hat{R}_i)^2, (\text{clip}(V_\phi(s_i), V_{\phi_{\text{old}}}(s_i) - \varepsilon, V_{\phi_{\text{old}}}(s_i) + \varepsilon) - \hat{R}_i)^2]]$, estimating state values that reflect progress toward both task completion and makespan minimization.

A key advantage of MAPPO is that while individual policies operate on local observations (enabling decentralized execution), the critic can leverage global information during training, making it more sample efficient [15]. The critic functions as a variance-reducing baseline that helps assess the quality of different action sequences leading to task completions, effectively guiding the policy toward solutions that satisfy our heterogeneous capability constraints while minimizing execution time. The fully-expanded MAPPO formulation, including actor and critic objectives, can be found in Appendix B.

4 Proposed Method

Our solution proposes an approach to model the behavior of non-communicating physical agents with unobservable intended goals. We employ the affordance model defined by Khetarpal et al., which formalizes the relationship between agent intent and environmental constraints. Within this framework, agent intent I_a constitutes the set of states an agent aims to reach through action $a \in A$ from state $s \in S$, while affordance represents the subset of states in I_a that an agent can successfully transition to via action a from state s.

The affordance algorithm in our methodology assumes I_a encompasses all available target positions. For agent a with distance function d, and state-action pairs (s, a) in the environment model M, the affordance algorithm $\mathbf{AFF}(s, a)$ approximates the true transition probability distribution $P(\cdot|s, a)$ such that $d(\mathbf{AFF}(s, a), P(\cdot|s, a)) \leq \epsilon$ for a small scalar value ϵ. In our task allocation problem, we estimate transition probabilities specifically to states containing targets.

Teammate behavior modeling enables efficient path planning toward attainable targets bound by an agent's capability constraints. To promote multi-agent collaboration in communication-constrained environments, we implement a global reward function that quantifies individual agent contributions to collective performance metrics, thereby incentivizing implicit coordination through environmental feedback mechanisms.

4.1 Computing Teammate Affordances

The teammate model proposed in our method uses the teammates' observed capabilities and spatial positions at each step of the episode and compares this information with each target's spatial position and requirements. The affordance computation algorithm 1 generates an affordance vector for each agent $i \in R$ representing the attractiveness of each target $j \in T$ in the environment. For an environment with $|T|$ targets, there is a $|T|$-dimensional affordance vector. While an agent i is computing its affordance score, it is considered the 'ego agent' and the other agents $k \in R \setminus \{i\}$ are considered 'teammate agents'. This affordance score is a combination of the ego agent's capability suitability ratio b_i^k / a_j^k to handle a target with a given set of requirements, and the agent-to-target distance $d(p_i, p_j)$ between the ego agent and the target. The affordance score of the ego agent is then weighted by the estimated affordances of the teammate agents that are near the ego agent. If the ego agent's teammates are more suitable to handle the target, then the affordance of the ego agent decreases. This re-weighting of the affordances of the ego agent serves as an anti-crowding technique that ensures that the agents are discouraged from heading towards the same targets. The concatenation of affordance values for each available target into an array creates the affordance vector. This vector is then normalized to produce a categorical set of values. The target with the highest value corresponds to the most afforded target by the ego agent.

Algorithm 1 begins by initializing key variables (lines 2-4), extracting the agent's capability b_i^k and position p_i while creating storage for target-specific affordance values α. For each target $j \in T$ (lines 5-29), the algorithm implements efficient filtering by skipping already visited targets (lines 6-7), gathering target-specific information (lines 8-9), and eliminating unsuitable matches where $b_i^k < a_j^k$ (lines 10-12). It then calculates a base affordance score α_j (lines 13-15) by combining the capability-requirement ratio β_{ij} with spatial distance metrics d_{ij}, creating an initial measure of target suitability. In lines 16-29, the algorithm analyzes the influence of other agents by adjusting affordances based on teammate positioning and capabilities: reducing values when more suitable agents are nearby ($\beta_{mj} > \beta_{ij}$) and slightly increasing affordances when the ego agent is better suited or incapable agents approach challenging tasks, indicating highly viable targets. After processing all targets, the algorithm normalizes these values (lines 31-32) to create a categorical probability distribution for decision-making purposes before returning the final affordance vector α (line 33), enabling agents to make informed, coordinated choices without explicit communication.

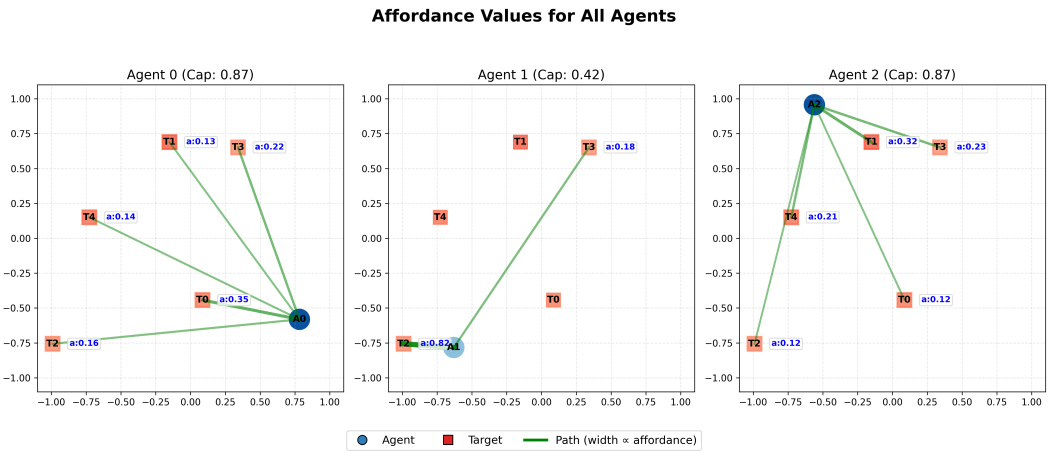


Figure 1: These images show the affordance values of 3 heterogeneous agents (0, 1, 2) in the same environment with heterogeneous targets. In each frame, we only display affordances and paths to targets the agent can afford (agent_capability $\geq$ target_requirements).

Algorithm 1: Affordance Computation Algorithm

Input: i: agent index, P: set of agent positions, B: set of agent capabilities, P_T: set of target positions, A: set of target requirements, V: set of visited targets

$\alpha \leftarrow \mathbf{0}_{|T|}$ // Initialize affordance vector

$b_i^k \leftarrow B[i]$ // Agent i's capability

$p_i \leftarrow P[i]$ // Agent i's position

for $j = 0$ **to** $|T| - 1$ **do**

  if $V[j]$ **then**

    continue // Skip visited targets

  if $b_i^k < a_j^k$ **then**

    continue // Skip if agent lacks capability

  $p_j \leftarrow P_T[j]$ // Target position

  $\beta_{ij} \leftarrow b_i^k / a_j^k$ // Capability ratio

  $d_{ij} \leftarrow \|p_i - p_j\|$ // Euclidean distance

  if $d_{ij} < \delta$ **then**

    // δ is a small constant

    $d_{ij} \leftarrow \delta$ // Avoid division by zero

  $\alpha_j \leftarrow \beta_{ij} / d_{ij}$ // Base affordance

  for $m = 0$ **to** $|R| - 1$ **do**

    // Check all teammates

    if $m = i$ **then**

      continue // Skip self

    $p_m \leftarrow P[m]$ // Teammate position

    $b_m^k \leftarrow B[m]$ // Teammate capability

    $d_{mj} \leftarrow \|p_m - p_j\|$ // Teammate-to-target distance

    if $d_{mj} < \rho$ ***and*** $b_m^k \geq a_j^k$ **then**

      // ρ is neighbor radius

      $\beta_{ij} \leftarrow b_i^k / a_j^k$ // Ego capability ratio

      $\beta_{mj} \leftarrow b_m^k / a_j^k$ // Teammate capability ratio

      if $\beta_{mj} > \beta_{ij}$ **then**

        $\alpha_j \leftarrow \alpha_j \times (d_{mj}/\rho) \times (\beta_{ij}/\beta_{mj})$ // Reduce affordance

      else

        $\alpha_j \leftarrow \alpha_j \times (d_{mj}/\rho) \times (1.0 + (\beta_{ij} - \beta_{mj}))$ // Adjust affordance

  $\alpha[j] \leftarrow \alpha_j$ // Store affordance value

$\gamma \leftarrow \sum_{j \in T} \alpha[j]$ // Sum of affordances

if $\gamma > 0$ **then**

  $\alpha \leftarrow \alpha / \gamma$ // Normalize to probability distribution

return α // Return affordance vector

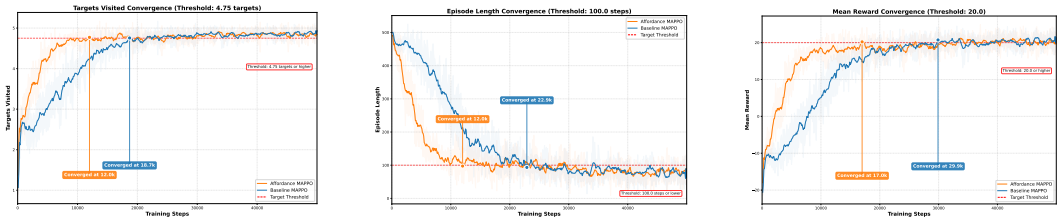
5 Experiments

Our MAPPO implementation incorporates an attention mechanism enabling agents to selectively focus on relevant inter-agent information, inspired by the performance gains established by Pu et al. [16]. Both actor and critic networks use a two-layer MLP structure with 64 hidden units per layer. Training hyperparameters include: learning rate: 0.0003, discount factor (γ): 0.99, GAE parameter (λ): 0.95, PPO clip ratio: 0.2, entropy coefficient: 0.01, batch size: 128, and 10 PPO epochs per batch. We adapted the gridworld environment from PettingZoo's Parallel Environment [17], which supports continuous movement with discrete actions. We modified the SimpleSpread environment to accommodate heterogeneous agents and tasks for our single-agent-multi-task assignment scenarios. In our configuration, $n \in R$ agents possess heterogeneous capabilities b_i^k uniformly sampled from $\mathcal{U}(0.3, 1.0)$, while $m \in T$ targets have requirement values a_j^k drawn from a similar distribution but

subject to additional constraints that guarantee at least one feasible allocation solution exists for each randomly generated problem instance. This ensures that each environmental configuration is solvable while maintaining the heterogeneous nature of the task allocation challenge. We evaluate performance using three complementary metrics: **sample efficiency**, which measures learning speed in terms of both training iterations and wall-clock time required to reach performance thresholds; **planning efficiency**, which quantifies the number of execution steps agents need to complete target visitation in a given instance; and **task completion rate**, which captures the percentage of targets successfully collected across environment instances.

We compare two MAPPO variants: (1) a baseline implementation and (2) our affordance-augmented MAPPO. We evaluate performance on instances not encountered during training and measure task completion rates, and planning efficiency across multiple unique evaluation episodes. All experiments were conducted on a workstation equipped with an NVIDIA RTX 3090 GPU (24GB VRAM) and a 32-core CPU with 128GB RAM.

6 Results



(a) Targets Visited (b) Episode Length (c) Rewards

Figure 2: Evaluation curves for both the baseline (blue) and the affordance-augmented (orange) MAPPO. As performance improves, execution steps decrease (shorter episodes) and more targets are completed per episode.

Figure 2 presents the training convergence metrics comparing Affordance-MAPPO against Baseline MAPPO across three random seeds. Across all three key performance indicators—target visitation rate, episode length, and mean reward—the affordance-augmented approach demonstrates significantly faster convergence while achieving comparable final performance.

As shown in Figure 2(a), Affordance-MAPPO reaches the target threshold of 4.75 visited targets after only 12,000 training steps, while Baseline MAPPO requires 18,700 steps, representing a 1.6× speedup (35.8% faster). Similarly, Figure 2(b) demonstrates that Affordance-MAPPO converges to the target episode length of 100 steps at 12,000 training steps, compared to 22,900 steps for Baseline MAPPO, yielding a **1.9× speedup** (47.6% faster). Perhaps most importantly, Figure 2(c) shows that Affordance-MAPPO achieves the target mean reward of 20.0 after 17,000 steps versus 29,900 steps for Baseline MAPPO—a **1.8× speedup** (43.1% faster).

Table 1: Wall clock time to convergence (averaged across 3 seeds)

Model	**Episode Length**	**Targets Visited**	**Mean Reward**
Affordance MAPPO	8.7h	7.9h	9.0h
Baseline MAPPO	26.5h	26.1h	26.5h
Speedup	**3.0×**	**3.3×**	**2.9×**

The sample efficiency improvements are even more pronounced when examining actual wall clock training time, as shown in Table 1. Affordance-MAPPO reaches the target episode length in just 8.7 hours of training, compared to 26.5 hours for Baseline MAPPO—a **3.0× reduction** in training time. Similarly, for target visitation and mean reward metrics, Affordance-MAPPO demonstrates **3.3×** and **2.9×** speedups, respectively.

We note that convergence thresholds were set independently for each performance metric based on task requirements: 4.75 targets represents 95% of the available targets, 100 steps reflects efficient task completion (approximately 20 steps per target), and a reward of 20.0 corresponds to the trade-off that maximizes successful completion with minimal step penalties. The variation in speedup factors across metrics (1.6× to 3.3×) reflects the different learning dynamics associated with each aspect of the task—with affordance augmentation particularly accelerating the learning of efficient trajectories and reward maximization behaviors.

These substantial improvements in real-world training time highlight the practical significance of affordance augmentation for robot learning applications, where computational resources and training time are often limited. The results strongly support our hypothesis that affordance-augmented observations provide a more informative state representation that accelerates policy learning. By implicitly encoding teammate modeling and task suitability information, the affordances effectively guide exploration toward promising regions of the policy space. Notably, both approaches eventually reach similar performance plateaus, indicating that affordance augmentation primarily benefits training efficiency rather than changing the fundamental capabilities of the underlying MAPPO algorithm.

6.1 Testing Performance on Unseen Scenarios

While our training convergence analysis demonstrates substantial efficiency improvements during the learning phase, it's equally important to evaluate how well the trained policies generalize to new scenarios and their computational requirements during execution. To assess this, we tested our trained models across 500 unique, unseen problem instances, measuring task performance and execution time overhead.

Table 2: Performance on 500 novel test scenarios (mean ± std across seeds)

Model	Target Rate	Avg Steps	Reward	Execution Time (s)
Affordance MAPPO	98.2% ± 1.4	40.6 ± 2.7	49.75 ± 1.19	0.072 ± 0.005
Baseline MAPPO	98.8% ± 0.4	39.9 ± 0.5	50.30 ± 0.31	0.067 ± 0.001

Table 2 presents the test results for our two primary models. Both approaches demonstrate excellent task completion capabilities, with target completion rates exceeding 98%. The Baseline MAPPO achieves slightly higher reward and marginally shorter episodes on average, though the differences fall within the standard deviation ranges. Most importantly, the execution time per episode is nearly identical between models (0.072s vs. 0.067s), with both achieving similar computational efficiency.

These results reveal an important insight: while affordance augmentation dramatically reduces training time (as shown in Table 1), it introduces negligible computational overhead during model execution. This confirms that the affordance mechanism primarily addresses the training bottleneck rather than affecting runtime performance. Once trained, both models generalize effectively to unseen task configurations with comparable performance profiles.

The minimal execution time difference (approximately 5ms per episode) between models is particularly noteworthy considering the substantial training time advantage of Affordance MAPPO. This suggests that in practical multi-robot applications, where training resources are often the limiting factor, affordance augmentation offers a compelling approach to accelerate learning without compromising deployment efficiency or generalization capabilities.

7 Conclusion

Our work demonstrates that augmenting agent observations with affordance values significantly improves both sample efficiency and wall clock training time in multi-robot task allocation problems. The affordance computation algorithm enables robots to implicitly model their teammates' behaviors and make more informed decisions without requiring explicit communication. The experimental results show that Affordance-MAPPO achieves 1.6×-1.9× faster convergence in terms of

training steps compared to the standard baseline, and even more dramatic improvements of 2.9×-3.3× in actual wall clock time—reducing training from over 26 hours to under 9 hours across all performance metrics.

This efficiency gain is particularly valuable in complex heterogeneous multi-robot teams, where training experiences are costly, coordination is challenging, and computational resources are limited. By providing a more structured observation space that incorporates capability matching and spatial relationships, affordances guide policy learning toward promising coordination strategies earlier in the training process. Importantly, this acceleration comes without sacrificing final performance, as both approaches eventually reach similar capability plateaus.

Future work should explore using affordance-augmented observations in more challenging environments with larger teams, higher target densities, and dynamic obstacles. Additionally, testing these approaches on physical robot systems with real-world sensing limitations and communication constraints would further validate the practical utility of affordance-based coordination for deployment in real-world multi-robot applications.

References

[1] S. Liu, W. Liu, W. Chen, G. Tian, J. Chen, Y. Tong, J. Cao, and Y. Liu. Learning multi-agent cooperation via considering actions of teammates. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[2] A. Agrawal, S. Hariharan, A. S. Bedi, and D. Manocha. Dc-mrta: Decentralized multi-robot task allocation and navigation in complex environments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11711–11718. IEEE, 2022.

[3] S. Amini, M. Palhang, and N. Mozayani. Pomcp-based decentralized spatial task allocation algorithms for partially observable environments. *Applied Intelligence*, 53(10):12613–12631, 2023.

[4] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, et al. Autort: Embodied foundation models for large scale orchestration of robotic agents. *arXiv preprint arXiv:2401.12963*, 2024.

[5] S. S. Kannan, V. L. Venkatesh, and B.-C. Min. Smart-llm: Smart multi-agent robot task planning using large language models. *arXiv preprint arXiv:2309.10062*, 2023.

[6] A. Thomas, F. Mastrogiovanni, and M. Baglietto. Towards multi-robot task-motion planning for navigation in belief space. *arXiv preprint arXiv:2010.00780*, 2020.

[7] X. Liu, P. Li, W. Yang, D. Guo, and H. Liu. Leveraging large language model for heterogeneous ad hoc teamwork collaboration. *arXiv preprint arXiv:2406.12224*, 2024.

[8] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[9] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti. Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.

[10] A. Skrynnik, A. Andreychuk, M. Nesterova, K. Yakovlev, and A. Panov. Learn to follow: Lifelong multi-agent pathfinding with decentralized replanning. In *PRL Workshop Series {\textendash} Bridging the Gap Between AI Planning and Reinforcement Learning*, 2023.

[11] T. T.-K. Lau and B. Sengupta. The multi-agent pickup and delivery problem: Mapf, marl and its warehouse applications. *arXiv preprint arXiv:2203.07092*, 2022.

[12] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[13] P. J. Gmytrasiewicz and P. Doshi. Interactive pomdps: Properties and preliminary results. In *International Conference on Autonomous Agents: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-*, volume 3, pages 1374–1375, 2004.

[14] C. Amato and F. Oliehoek. Scalable planning and learning for multiagent pomdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[15] 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*, 35:24611–24624, 2022.

[16] Z. Pu, H. Wang, Z. Liu, J. Yi, and S. Wu. Attention enhanced reinforcement learning for multi agent cooperation. *IEEE Transactions on neural networks and learning systems*, 34(11): 8235–8249, 2022.

[17] J. K. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sulivan, L. Santos, R. Perez, C. Horsch, C. Dieffendahl, N. L. Williams, Y. Lokesh, and P. Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2021.

[18] H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926, 2009.

A Component Measures for Computing Affordances

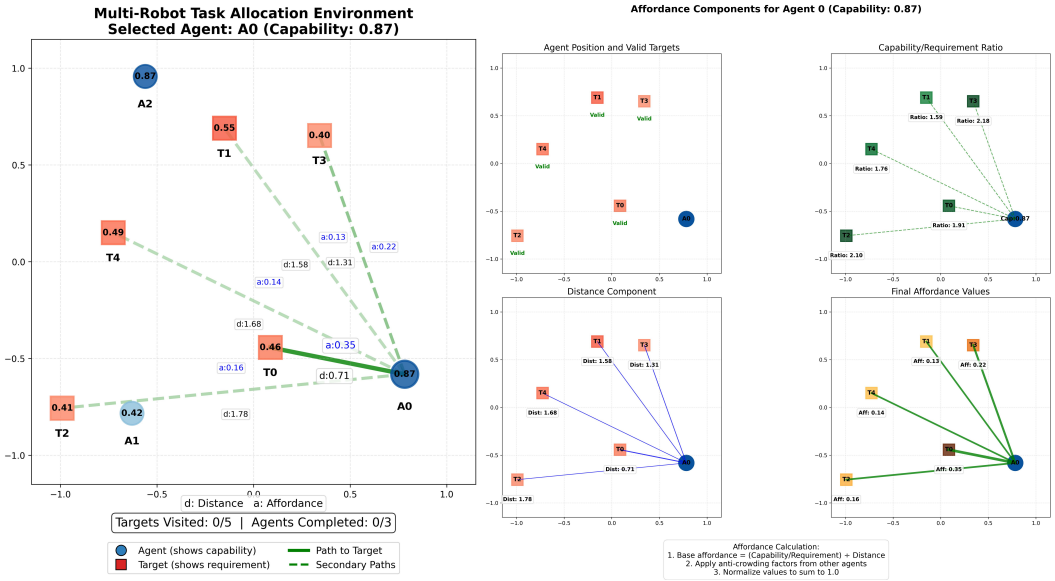


Figure 3: The image on the left shows the optional paths Agent 0 can take to get to any of the targets it can afford. The images on the right capture the various components that go into computing the affordances of various targets relative to the ego agent: agent positions, relative distances, and the agent's capability to target requirement ratio.

B MAPPO Formulation

MAPPO is made up of an **actor** network (policy function) and a **critic** network (value function). These two networks are trained with specific objectives [15]:

B.1 Actor Objective

The actor network is trained to maximize:

$$L(\theta) = \mathbb{E}\left[\min(r_{\theta,i} \cdot A_i, \text{clip}(r_{\theta,i}, 1-\varepsilon, 1+\varepsilon) \cdot A_i)\right] + \sigma \cdot \mathbb{E}[S[\pi_\theta(o_i)]] \tag{5}$$

Where:

- $r_{\theta,i} = \frac{\pi_\theta(a_i|o_i)}{\pi_{\theta_{\text{old}}}(a_i|o_i)}$ is the importance sampling ratio
- A_i is the advantage estimate computed using GAE (Generalized Advantage Estimation)
- S is the policy entropy
- σ is the entropy coefficient
- ε is the clipping parameter that limits policy updates

B.2 Critic Objective

The critic network is trained to minimize:

$$L(\phi) = \mathbb{E}\left[\max\left[(V_\phi(s_i) - \hat{R}i)^2, (\text{clip}(V\phi(s_i), V_{\phi_{\text{old}}}(s_i) - \varepsilon, V_{\phi_{\text{old}}}(s_i) + \varepsilon) - \hat{R}_i)^2\right]\right] \tag{6}$$

Where $\hat{R}_i$ represents the discounted cumulative reward estimate. By optimizing these objective functions, MAPPO learns policy parameters θ^* and value function parameters ϕ^* that maximize expected returns while maintaining learning stability through PPO's trust region constraints (implemented via the clipping mechanism). The critic network serves as a baseline that reduces variance and evaluates long-term action sequences, guiding policy updates toward more effective task allocation strategies. Meanwhile, the centralized training framework enables the critic to process global state information during training, producing more accurate value estimates than would be possible with purely decentralized learning, while preserving the ability to execute policies based solely on local observations.

C Reward function incorporating global rewards

In a bid to encourage teammate collaboration, we proposed using shared rewards reflective of an agent's contribution to the team's success. The reward function described in Algorithm 2 implements a multi-component reward structure that balances individual achievement with team performance. At each timestep, an agent $i \in R$ receives a navigation penalty $c_{\text{step}} = -0.05$ for taking a navigation action (line 1). The algorithm then identifies each agent's set of affordable targets $\mathcal{T}_i$ based on capability constraints $b_i^k \geq a_j^k$ (line 3), implementing the constraint from Equation 1.

When an agent approaches a target within distance threshold $\delta = 0.1$ (line 8) that it can afford and hasn't been visited, it receives a significant reward $c_{\text{completion}} = +10$ (line 10) that offsets the navigation penalty, and the target is marked as visited (line 11). This aligns with our makespan objective in Equation 3 by incentivizing efficient task completion. If an agent visits an affordable target that has already been serviced by another agent, it incurs a moderate penalty $c_{\text{revisit}} = -0.1$ (line 14), encouraging agents to take efficient paths to available targets that their teammates are unlikely to reach first.

When the global reward mechanism is activated (lines 19-26), agents receive partial rewards based on the team's overall performance, calculated as a function of the target completion ratio ρ (line 20). This encourages agents to consider the entire team's progress when making decisions. Finally, when all tasks are completed before the episode terminates (lines 28-33), each agent receives a bonus $c_{\text{bonus}} = +5$ scaled by its individual contribution ratio γ_i (lines 30-31), implementing a fair reward distribution that satisfies our target completion objective in Equation 4.

Algorithm 2: Reward Function with Global Rewards

Input: P: agent positions, P_T: target positions, V: visited targets, D: distances

$r \leftarrow \{i : c_{\text{step}} \mid \forall i \in R\}$; // Initialize with step penalties

for $i \in R$ **do**

  $\mathcal{T}_i \leftarrow \{j \in T \mid b_i^k \geq a_j^k\}$; // Valid targets for agent i

  if $\mathcal{T}_i = \emptyset$ **then**

    continue ; // Skip if no valid targets

  end

  $j^* \leftarrow \arg\min_{j \in \mathcal{T}_i} \|p_i - p_j\|$; // Closest valid target

  $d_{ij^*} \leftarrow \|p_i - p_{j^*}\|$; // Distance to closest target

  if $d_{ij^*} < \delta$ **then**

    if $\neg V[j^*]$ **then**

      $r[i] \leftarrow c_{\text{completion}}$; // Reward for task completion

      $V[j^*] \leftarrow \texttt{true}$; // Mark target as visited

      $C[i] \leftarrow C[i] + 1$; // Increment completed targets count

    end

    else

      $r[i] \leftarrow c_{\text{revisit}}$; // Penalty for visiting completed target

    end

  end

end

if *use_step_bonus_rewards* **then**

  $\rho \leftarrow \frac{\sum_{j \in T} V[j]}{|T|}$; // Target completion ratio

  $r_{\text{team}} \leftarrow 0.2 \times \rho$; // Team reward based on completion

  for $i \in R$ **do**

    if $\neg$ *terminated*$[i]$ **then**

      $r[i] \leftarrow r[i] + r_{\text{team}}$; // Add team reward component

    end

  end

end

$\texttt{all\_completed} \leftarrow \bigwedge_{j \in T} V[j]$; // Check if all targets visited

if *all_completed* $\wedge$ *_use_global_bonus_rewards* **then**

  for $i \in R$ **do**

    $\gamma_i \leftarrow \frac{C[i]}{|T|}$; // Agent's contribution ratio

    $r[i] \leftarrow r[i] + c_{\text{bonus}} \times \gamma_i$; // Add weighted bonus

  end

end

return r ; // Return reward vector

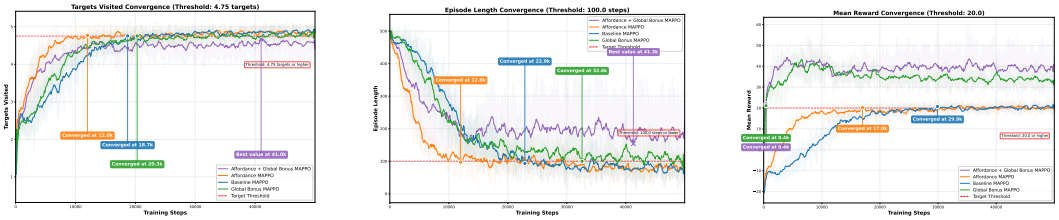
D Training Performance of MAPPO with Affordances and Global Rewards

In this experiment, we evaluate how two key factors affect multi-robot task allocation performance: observation information and reward structure. We test these combinations across multiple random seeds to ensure reliable results. For observation types, we compare (1) Baseline Observation—a tensor containing agent positions, target positions, agent capabilities, target requirements, ego-agent attributes, and target visitation status, and (2) Affordance-Augmented Observation—which extends the baseline with an affordance map from Algorithm 1, providing a t-dimensional tensor showing the viability of each target from the agent's perspective. For reward structures, we implement two approaches: (1) Standard Rewards—where agents receive normal physical action penalties and completion bonuses based on their performance when all tasks are completed before the episode ends, and (2) Global Team Rewards—which adds team-wide completion incentives to the standard rewards, where all agents receive rewards based on the team's overall performance.

By combining these observation types and reward structures, we create four distinct model variants for our analysis:

- **Baseline**: Standard observations with standard rewards
- **Affordance**: Affordance-augmented observations with standard rewards
- **GlobalReward**: Standard observations with global team rewards
- **GlobalReward+Affordance**: Affordance-augmented observations with global team rewards

Additional variants incorporating a step-bonus reward structure were initially evaluated but showed performance characteristics that warranted separate analysis, which we present in Section D.1.1.



(a) Average Targets Visited (b) Average Episode Length (c) Average Rewards

Figure 4: These plots represent the evaluation results over the course of 50K training iterations. We compare the 4 variants introduced in Section D

D.0.1 Training Convergence Analysis

Figure 4 presents a comprehensive evaluation of our four MAPPO variants across key performance metrics during 50K training iterations. These results offer valuable insights into how affordance augmentation and reward structures affect learning efficiency and final performance.

In terms of target visitation (Figure 4a), Affordance MAPPO demonstrates superior sample efficiency, reaching the threshold of 4.75 targets at just 12,000 steps—significantly faster than Baseline MAPPO (18,700 steps) and Global Bonus MAPPO (20,300 steps). This represents a 1.6× speedup over the baseline approach. Notably, the Affordance + Global Bonus MAPPO variant never consistently reaches the target threshold despite initial promising performance, plateauing around 4.5 targets even after 41,000 steps.

The episode length convergence (Figure 4b) shows even more dramatic differences. Affordance MAPPO reaches the target threshold of 100 steps in just 12,000 training iterations, while Baseline MAPPO requires 22,900 steps (1.9× slower) and Global Bonus MAPPO needs 32,600 steps (2.7× slower). The Affordance + Global Bonus variant exhibits particularly unstable behavior, with episode lengths remaining consistently higher than other variants throughout training, only reaching its best value of 153.8 steps at 41,300 steps—still above our target threshold.

The mean reward comparison (Figure 4c) reveals an interesting pattern where both global bonus variants immediately achieve high rewards due to their augmented reward structure, converging to the 20.0 threshold after only 400 steps. However, this early convergence is primarily an artifact of the reward implementation rather than reflecting genuine performance improvement. Between the standard reward variants, Affordance MAPPO reaches the target reward threshold at 17,000 steps compared to Baseline MAPPO's 29,900 steps—a substantial 1.8× speedup.

These results, summarized in the convergence analysis table, indicate that while affordance augmentation significantly improves learning efficiency (with speedups between 1.6× and 1.9× across metrics), combining affordances with global bonus rewards creates learning instability. This suggests potential conflicts between explicit reward-based cooperation signals and implicit affordance-based coordination. Based on these findings, we decided against using the global bonus variants

for subsequent experiments despite their initially promising reward profiles, as they either underperformed (Affordance + Global Bonus) or converged substantially slower (Global Bonus) on actual performance metrics.

D.1 Comparative Analysis of Model Variants

To provide a more comprehensive understanding of model performance, we extended our evaluation to include all four model variants on the 500 unseen test scenarios. The results of this analysis are summarized in Table 3.

Table 3: Performance comparison of all model variants on test scenarios

Model	Target Rate (%)	Avg Steps	Execution Time (s)
Affordance MAPPO	98.2 ± 1.4	40.6 ± 2.7	0.072 ± 0.005
Baseline MAPPO	98.8 ± 0.4	39.9 ± 0.5	0.067 ± 0.001
Aff+GlobalBonus MAPPO	91.4 ± 2.0	57.3 ± 5.9	0.097 ± 0.009
GlobalBonus MAPPO	71.7 ± 3.8	85.7 ± 3.6	0.129 ± 0.005

While we exclude direct reward comparisons due to the different reward structures between models, the operational metrics reveal clear performance patterns. The models without global bonus rewards (Affordance MAPPO and Baseline MAPPO) achieve significantly higher target completion rates and require fewer steps to complete episodes. This performance advantage also translates to faster execution times, with both standard models processing episodes nearly twice as quickly as the GlobalBonus MAPPO variant.

Notably, while affordance augmentation substantially improves training efficiency for both reward structures, it appears to have different effects on generalization depending on the reward formulation. With standard rewards, affordance-augmented and baseline models perform similarly on test scenarios. However, with global bonus rewards, affordance augmentation provides a considerable boost in target completion rate (91.4% vs. 71.7%) and execution efficiency (57.3 vs. 85.7 steps), suggesting that affordances may help stabilize learning when using more complex reward structures.

D.1.1 Emergent Trend: Decreasing Reward Despite Improved Performance

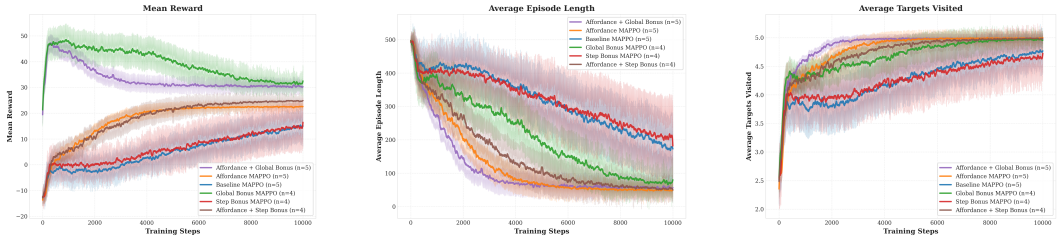
During our initial experiments, we implemented two team-based reward strategies intended to encourage cooperation:

- *Step bonus*: A small incremental reward awarded each timestep based on each agent's individual contribution to task completion.
- *Global reward*: Combines the step bonus with an additional team-wide completion bonus that is distributed equally among all agents when all targets are successfully visited.

We discovered an unintended anomaly in our reward implementation: as agents improved their performance during training, their cumulative rewards actually decreased over time. This counterintuitive trend is clearly visible in Figure 5. While Figure 5a shows steadily declining rewards for the global and step bonus models, Figure 5b confirms these same agents were completing episodes in progressively fewer timesteps, and Figure 5c verifies they were successfully visiting more targets.

This anomaly occurred due to a logical implementation error in our reward structure. As agents became more efficient, completing tasks in fewer steps, they had fewer opportunities to accumulate the per-step bonuses. The completion reward for a target (calculated as +10 - ((0.05 + step_bonus) $\times$ steps_taken)) remained relatively constant, but the total step bonuses decreased as episodes shortened.

Due to this unintended behavior, we conducted subsequent experiments without the step bonus mechanism, focusing instead on our four primary model variants with properly structured rewards to ensure that improvement in performance corresponded appropriately with reward signals.



(a) Average Rewards (b) Episode Length (c) Targets Visited

Figure 5: Training plots for MAPPO variants over 10K steps. As performance improves (shorter episodes, more targets), average episode rewards drop due to fewer steps available to accumulate per-step bonuses.

E Evaluating the Generalization Capability of an Affordance Heuristic MRTA Algorithm

The generalization results show that the models struggle with generalized environments, solving the same configuration. It did not make sense to evaluate the PPO models on varying scales, therefore, I decided to design a simple heuristic algorithm that used the affordance algorithm 1 to select the best target. The best target for an agent is the agent with the highest affordance. All these MRTA algorithms use A-star for path planning. I compare this Affordance Heuristic with two baselines:

- A **greedy baseline** that selects the closest target that an agent can afford (agent capability $\geq$ target)
- **Consensus Based Bundling Algorithm (CBBA)** [18] that provides a decentralized bidding mechanism for agents to select tasks and maximize agent utility

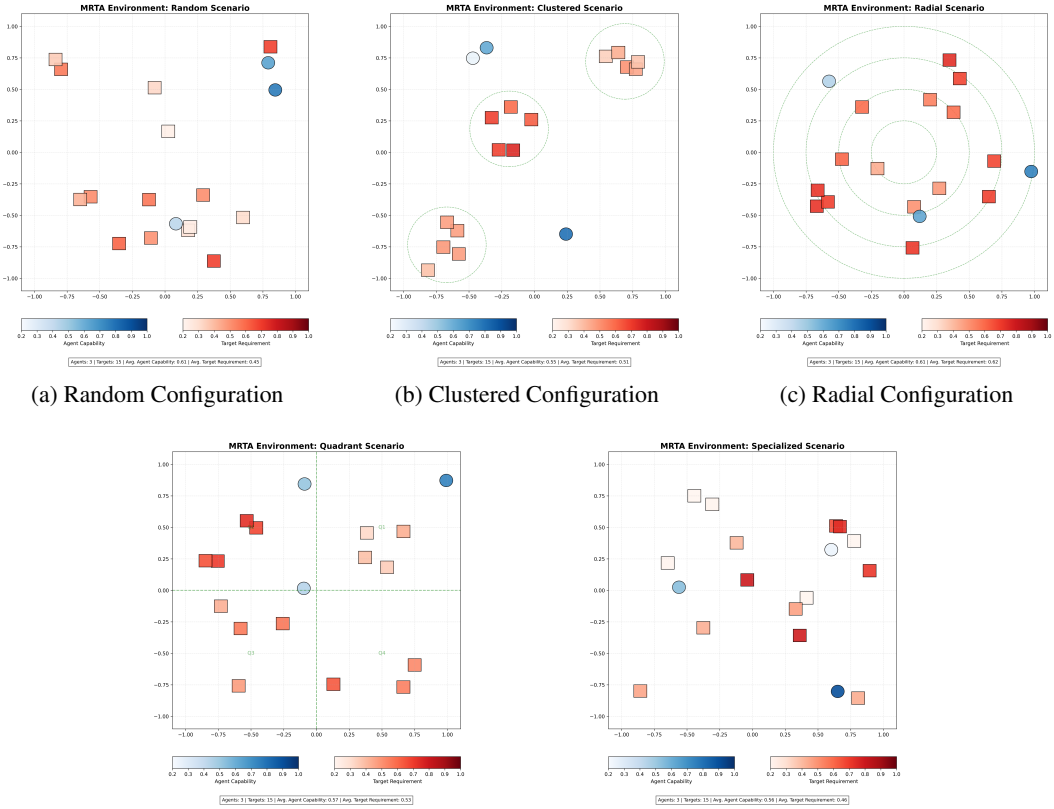
I test these algorithms on different levels of target congestion: small_scale, medium_scale, and large_scale and different target configurations as seen in figure 6

Within the clustered 6b and quadrant 6d configurations, targets with requirements beyond a specific threshold could be bundled together to increase the complexity of the environment or the targets could be randomly placed in clusters, making it difficult for a single agent to complete all tasks in one go.

Table 4: Summary of Multi-Agent Coordination Benchmark Experiments

Experiment Name	**Agent–Target Configurations**	**Target Range**
Agent-Scale	15 agents $\times$ $\{5, 10, 15\}$ targets	5–15
Equal Assignment	N agents $\times$ N targets, $N \in \{3, 5, 6, 10, 12\}$	3–15
Generalization-Scale	10 agents $\times$ $\{5, 7, 10, 12, 15\}$ targets	5–15
Target-Scale	5 agents $\times$ $\{5, 7, 10, 12, 15\}$ targets	5–15

Each experiment in table 4 is designed to test a distinct aspect of multi-agent coordination. The **Agent-Scale** experiment evaluates how algorithms handle crowding by fixing the number of agents at 15 and scaling the number of tasks, revealing behaviors under contention. The **Equal Assignment** experiment explores balanced configurations where the number of agents matches the number of targets. This setup highlights precise coordination and full task coverage as the system scales. The **Proportional-Scale** experiment keeps the agent count constant at 10 while varying target counts to assess robustness to task density shifts. Lastly, the **Target-Scale** experiment fixes the agent team at 5 and increases the number of targets, testing each algorithm's ability to scale up to higher task demands and avoid redundancy or overload.



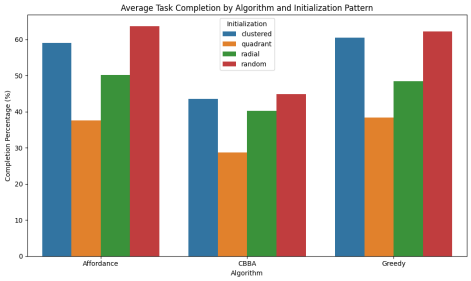
(a) Random Configuration (b) Clustered Configuration (c) Radial Configuration

(d) Quadrant Configuration (e) Specialized Configuration

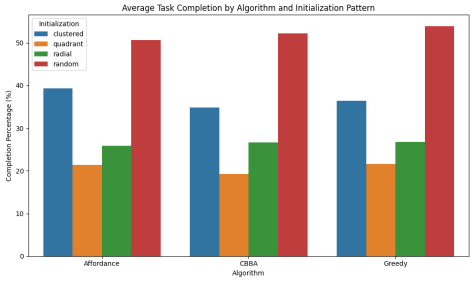
Figure 6: These plots show all 5 target initialization patterns. The blue scale represents the agent capabilities from least capable (light blue) to most capable (dark blue). The red scale represents target requirements from least requirements (light red) to most requirements (dark red)

The Affordance algorithm generalizes well to different initialization patterns and varying agent-to-target densities. The results in Figure 7 show that it might be too similar to the greedy algorithm, as it fails to show significant improvement even when tested at greater scales, referenced in Table 4. For these experiments, all algorithms maxed out on the experiment steps and did not achieve 100% target completion because the grids were intentionally large and target-dense to create complex scenarios. Therefore, we measure throughput across the different MRTA algorithms.

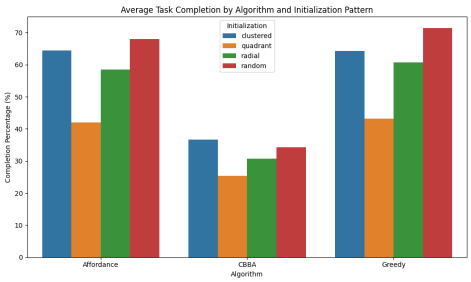
Our affordance-based heuristic task planner exhibits modest but consistent improvements over greedy algorithms (0.7-2.2%), while achieving substantially higher task completion rates against CBBA implementations (3.2-9.6%) across varied target initialization patterns in dense environments. The primary contribution is a generalizable affordance computation function that functions effectively as either an independent task planner or as an observation enhancement for policy training in decentralized, communication-free multi-robot task allocation scenarios.



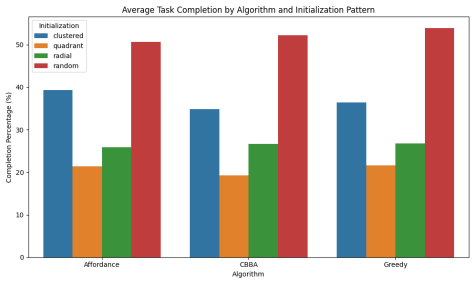
(a) Equal Agents & Targets



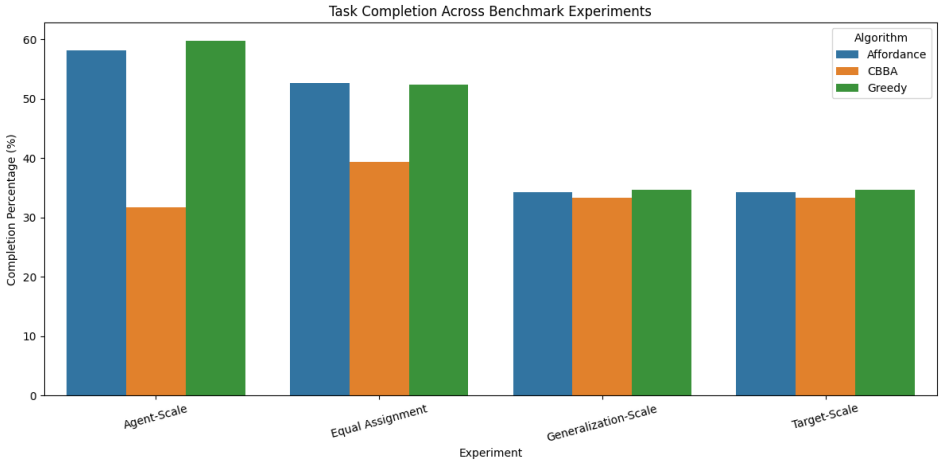
(b) Scaling Targets



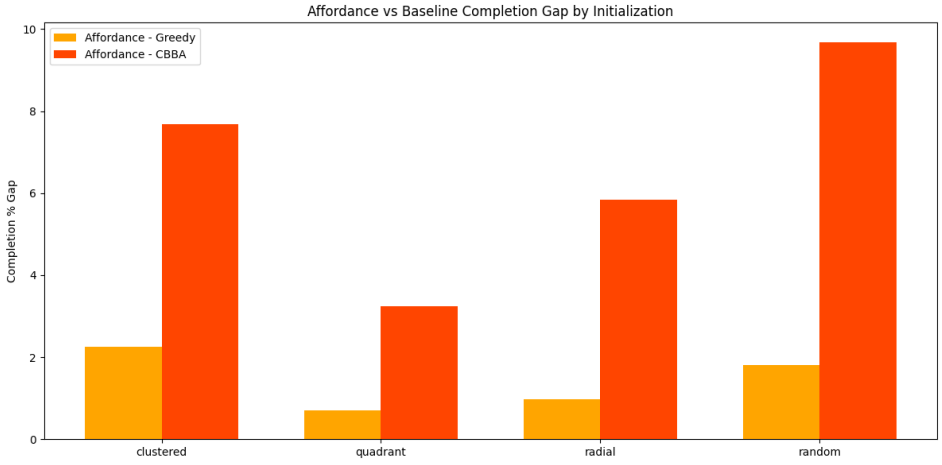
(c) # Targets > # Agents



(d) Proportional Scaling: Agents & Targets



(e) Overall Task Completion Summary Across All Experiments



(f) Performance Gap Between Affordance Heuristic and Greedy and CBBA Baselines

Figure 7: Task completion results across all 4 experimental configurations. The final plot summarizes average target completion across all 4 experiments and 3 MRTA algorithms.