

Leave It Better Than Found: Value-Guided Cooperation in Sequential Tasks

Mario Gomez Andreu^{1,2}
margomez@ethz.ch

Michael Amir¹
ma2151@cam.ac.uk

Antonio Marino¹
am3507@cam.ac.uk

Tianxu An²
tianan@ethz.ch

Marco Hutter²
mahutter@ethz.ch

Amanda Prorok¹
asp45@cam.ac.uk

Abstract:

We consider the problem of *sequential task continuation*, where one robot acts in a workspace that has been affected by a previous robot. Current approaches in multi-robot cooperation rely heavily on co-training or co-optimization, assuming that the robots have access to collective interaction data during the policy training or planning process. However, as increasingly varied robotic embodiments and policies are deployed and operate in the same real-world environment, learning policies for cooperation becomes infeasible due to the combinatorial explosion of possible robot combinations that need to be accounted for. To address this challenge, this paper considers the problem of *ad hoc* sequential cooperation between robots that have never before operated together in a shared environment. We demonstrate that robots can explicitly communicate their environmental preferences through sharing their *value functions*, enabling coordination without the need for prior joint training. To ground this exchange and prevent overconfidence in out-of-distribution states, we further introduce a k -nearest neighbors (KNN)-based uncertainty filter. Our method yields an increase of up to 48.5 pp compared with naive or even selfish agents, in task success rates across simulated and real-world experiments, demonstrating that value function transmission between agents is useful not merely as an internal learning tool, but as an effective basis for ad hoc coordination.

Website: <https://proroklab.github.io/PAVE/>

Keywords: Multi-Agent Systems, Sequential Tasks, Ad-Hoc Cooperation, Online Adaptation, Improvised Cooperation

1 Introduction

As robotic systems achieve greater autonomy, heterogeneous agents, e.g., cleaners, delivery bots, and security systems, will increasingly operate in the same public and domestic spaces. Robots modify these spaces to achieve their goals, leaving subsequent actors with the environments that they left behind. These workspace overlaps require spontaneous cooperation between independently developed systems. We formalize this challenge as *sequential task continuation*: a setting where an agent must proactively anticipate how its physical actions will alter the shared space and affect subsequent actors.

Currently, robot behaviors are typically designed in isolation by different parties to perform distinct tasks. When actors interact in the same shared workspace, we require methods and protocols

¹Prorok Lab, Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom ²Robotic Systems Lab, Department of Mechanical and Process Engineering, ETH Zürich, Zürich, Switzerland

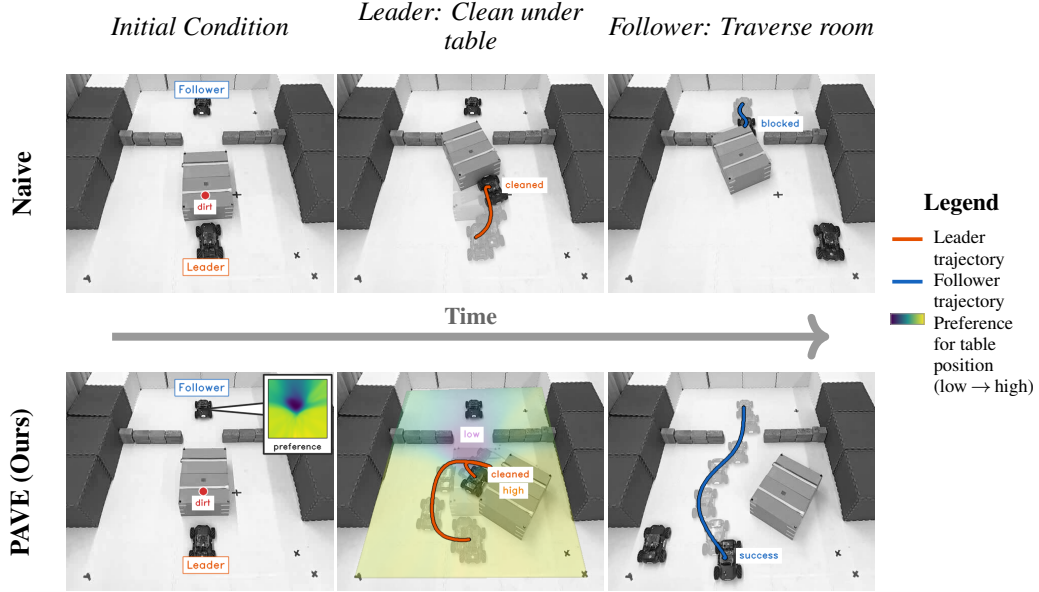


Figure 1: Two agents perform tasks sequentially in a shared environment: a leader pushes the table aside to clean beneath it; a follower must then traverse the room. **Naive** (top), the leader is unaware of the follower’s needs and leaves the table blocking the path, preventing traversal. **PAVE (Ours)** (bottom), the follower shares its value function over environment states, and the leader adapts by placing the table in a high-value region (overlaid with heatmap) for the follower, clearing the path, and enabling traversal.

that bridge these independent systems. For instance, delivery bots should not drop packages along common patrol routes, and cleaners should not rearrange the environment in such a way that areas become inaccessible to other robots. Because hand-crafted rules are too brittle to handle the infinite edge cases of open-world environments, learning-based coordination is required. Training a separate cooperative policy for every possible pairing of robots is unrealistic, as the possible combinations increase exponentially with the number of robot embodiments and tasks.

Instead, we turn to *ad hoc* coordination between agents that does not require joint training, as robots can easily share key components of their decision-making with other agents. Given the representation of other agents’ preferences, robots can adapt their behavior on the fly (*ad hoc*) to aid other robots in their tasks while still pursuing their own. Existing frameworks in this domain, however, rely on restrictive assumptions. Zero-shot coordination [1] trains a single robust policy, but it does not allow online adaptation. Ad Hoc teamwork, first proposed in Stone et al. [2] and implemented in PLASTIC [3] and TEAMSTER [4], adapts online but assumes a single common payoff and infers teammate behavior from observation rather than direct communication. Both approaches assume that actors act simultaneously and not sequentially.

Our approach is different: agents pursue distinct tasks and exchange compact representations of their preferences. Rather than relying on task-specific learned communication channels [5, 6], we leverage a quantity that reinforcement-learning agents already learn: the value function, which maps states to expected return. For *sequential task continuation*, the relevant part is the follower’s value over shared environment states. Sharing the value function tells the leader which handoff states are useful for the follower. Our method, **Policy Adaptation via Value Exchange (PAVE)**, uses this single exchange of the value function to coordinate with the following agent without joint training, adapting the leading agent’s policy online using a receding horizon planner.

We contribute:

1. The formalization of *sequential task continuation* with environment handoff.

2. PAVE: A method that combines value-function sharing with an online receding-horizon planner to coordinate two independently trained robots.
3. An uncertainty-based grounding using a k -nearest-neighbors distribution filter to prevent the receiving agent from acting on unreliable value estimates.

We empirically show an average 27.9 pp improvement in follower task success over a non-cooperative baseline (up to 48.5 pp in the most-constrained case) across five simulated scenarios and a real-world deployment.

Related Work Multi-agent reinforcement learning typically relies on joint training with shared critics, as in MAPPO [7], MADDPG [8], COMA [9], and QMIX [10]. These approaches do not scale to *sequential task coordination* as the robots act not simultaneously but sequentially in a shared environment. Cooperative Consecutive Policies [11] instead train distinct policies acting on a shared embodiment to solve long-horizon tasks, by optimizing each current policy with respect to the next agent’s Q-function to ensure an optimal hand-off. Although distinct from our method, it assumes a shared embodiment and joint training, and it inspires one of our baselines. Dhakal et al. [12] propose a planner for rearrangement tasks that anticipates the effects of its actions on future tasks of the same agent. PAVE differs by reusing an existing value function, filtering out-of-distribution value estimates, and considering multiple agents instead of a single agent.

Other approaches similarly treat the environment as the medium of inter-agent influence. Stigmergy [13, 14, 15] is a natural process used by swarm animals such as ants to coordinate through pheromone deposits in the environment. Environment Co-Design [16, 17, 18] also considers adapting the environment to benefit the agents acting within it. PAVE defines a middle ground: agents communicate directly via their value functions, but enact that coordination indirectly. One agent manipulates the physical space via its task-constrained policy to favorably shape the state for the next agent, achieving continuation without requiring central design or shared training. For an extended discussion of the related literature, we refer the reader to Appendix B.

2 Problem Statement

Consider the HOUSEHOLD scenario in Figure 1. A cleaning robot has to move a table to clean below it. Unaware of other actors in the environment, it might reposition the table in front of a pathway, blocking another robot (that does not have the ability to move the table) from subsequently traversing the room. If the following agents’ preferences for the environment are considered, the cleaning robot can still succeed while removing the table from the traversing agent’s path and *leaving the environment better than it found it*.

Setting. More generally, we consider cooperation problems in which agents act sequentially in the same environment. We call the first agent the leader L and the second agent the follower F . Both share a common representation of their environment $\mathbf{w}$ as well as individual private states $\mathbf{s}^L$ and $\mathbf{s}^F$. They act according to policies π_L and π_F with reward functions r_L, r_F . Their expected returns are $J_L(\pi_L) = \mathbb{E}_{\mathbf{s}_0^L \sim \rho_L, \mathbf{w}_0 \sim \rho_W} \left[\sum_{t=0}^T \gamma^t r_L(\mathbf{s}_t^L, \mathbf{w}_t, \pi_L(\mathbf{s}_t^L, \mathbf{w}_t)) \right]$, $J_F(\pi_F, \mathbf{w}) = \mathbb{E}_{\mathbf{s}_0^F \sim \rho_F} \left[\sum_{t=0}^K \gamma^t r_F(\mathbf{s}_t^F, \mathbf{w}_t, \pi_F(\mathbf{s}_t^F, \mathbf{w}_t)) \mid \mathbf{w}_0 = \mathbf{w} \right]$ where ρ_L and ρ_F are the initial distributions of the leader’s and follower’s private states, respectively, ρ_W is the initial distribution of the environment state, and $\gamma \in (0, 1]$ is the discount factor. The follower’s expectation conditions on $\mathbf{w}_0 = \mathbf{w}$, i.e. the handoff state passed in from the leader.

Sequential Subtask Execution. The leader acts in the environment for T steps leaving behind a terminal environment state $\mathbf{w} \sim H_T(\pi_L)$ distributed according to the handoff distribution H_T , which depends only on the leader policy π_L and world dynamics.

Cooperative Leader. To achieve meaningful cooperation, the leader cannot strictly maximize its own return, J_L . It must concurrently optimize for the follower’s expected return given the inherited environment state. Because the leader’s influence on the follower is solely dictated by the handoff

distribution, H_T , the joint cooperative objective becomes

$$\max_{\pi_L} J_L(\pi_L) + \alpha \mathbb{E}_{\mathbf{w} \sim H_T(\pi_L)} [J_F(\pi_F, \mathbf{w})]. \quad (1)$$

with weight α . The follower policy is assumed to be fixed, reacting only to the handoff state that the leader leaves behind.

Modular Composition As this work focuses on ad hoc coordination, we assume that collective interaction data is not accessible prior to deployment. This requires the leader to solve (1) on-line during deployment. Further, the leader needs to evaluate the follower’s performance J_F under arbitrary environment states $\mathbf{w}_T$, requiring detailed knowledge about the follower’s behavior, its interaction dynamics with the environment, and its reward function to simulate its response to the leader’s behavior when planning. We address this problem in the following section by introducing a simplified preference interface that abstracts this complexity and is easy to obtain.

3 Method

Modularity stems from abstraction through common interfaces. We propose using the value function as an abstraction for coordination. The value function of a policy maps states to expected returns under that policy. Because agents in a shared environment operate on a common environment state $\mathbf{w}$, the *environmental value function* $V_F(\mathbf{w})$ naturally serves as a preference function, communicating how desirable a given environment state is to the follower. By construction, $V_F(\mathbf{w}) \doteq J_F(\pi_F, \mathbf{w})$, meaning it equals the follower’s expected return from the handoff state $\mathbf{w}$. Substituting into (1), the cooperative objective becomes

$$\max_{\pi_L} J_L(\pi_L) + \alpha \mathbb{E}_{\mathbf{w} \sim H_T(\pi_L)} [V_F(\mathbf{w})]. \quad (2)$$

By revealing its value function, the follower enables the leader to optimize (2) on its behalf without requiring any further input. In practice, we obtain the environmental value function by aggregating the critic of an agent trained with PPO [19]. The critic $V_F^{\text{critic}}(\mathbf{s}, \mathbf{w})$ depends on the full follower state $(\mathbf{s}, \mathbf{w})$, comprising the follower’s private state $\mathbf{s}$ and the environmental state $\mathbf{w}$. Since the leader optimizes only over $\mathbf{w}$, we aggregate the critic to produce the environmental value function — either by fixing the initial private state to a known value $\mathbf{s}_0$ ($V_F(\mathbf{w}) = V_F^{\text{critic}}(\mathbf{s}_0, \mathbf{w})$), or by taking the expectation over the follower’s initial private state distribution $V_F(\mathbf{w}) = \mathbb{E}_{\mathbf{s}_0 \sim \rho_F} [V_F^{\text{critic}}(\mathbf{s}_0, \mathbf{w})]$. In this work, we only consider the case of a known initial state for the follower. Because PAVE shares only V_F , the leader cannot recover π_F without an additional model of the follower’s private state and dynamics. Specifically, V_F does not expose the follower’s individual state or action distribution.

Cooperative Objective Combining two qualitatively different objectives is nontrivial and domain-dependent. The proposed approach allows for the selection of arbitrary aggregation functions. In this work, we combine the two objectives by first normalizing them and then summing them using a weighting factor, α .

$$\max_{\pi_L} \frac{J_L - J_L^{\min}}{\Delta J_L} + \alpha \frac{V_F - J_F^{\min}}{\Delta J_F} \quad (3)$$

where $J_L^{\min}, \Delta J_L, J_F^{\min}, \Delta J_F$ are determined from rollouts of the leaders’ and followers’ prior policies.

Uncertainty-Aware Value Interface Because the leader and follower are not trained jointly, the environment states generated by the leader may shift away from those seen during the follower’s training. This distribution shift substantially reduces the accuracy of out-of-distribution value estimates. It is therefore necessary to estimate value uncertainty and adapt the leader’s objective accordingly. Standard uncertainty quantification methods typically require modifying the follower’s architecture or training pipeline, which violates the decoupled nature of ad hoc cooperation. Instead, we adopt a heuristic that can be evaluated directly from the follower’s training-state distribution: a

KNN-based approach to determine whether a handoff state is out of distribution, similar to [20]:

$$\beta(\mathbf{w}_T) = \text{clip}\left(0, 1, \frac{\frac{1}{K} \sum_{k=1}^K \|\text{neighbor}(\mathbf{w}_T, k) - \mathbf{w}_T\|_2 - \tau}{d_{max} - \tau}\right) \quad (4)$$

$\beta(\mathbf{w}_T)$ is close to zero when environment states are near the training samples on which the follower was trained, and becomes large when they are far away. This can be leveraged to adjust the value function in regions of the state space where the agent, and therefore its value function, has not been trained. We choose $K = 5$. The parameters τ and d_{max} are chosen to be, respectively, the 95th and 99.9th percentiles of the empirical distribution of the mean distance to the K nearest neighbors of each state, computed over a sample of 1000 states drawn from the environmental state distribution on which the follower was trained.

Cooperative Online Planning To enable ad hoc cooperation, the planning procedure must be computationally feasible online. Because trajectory optimization benefits significantly from a strong prior, we use the leader’s independently trained policy π_L to warm start the search. At each step, the planner seeks to maximize the normalized, uncertainty-aware cooperative objective:

$$\max_{\pi_L} \frac{J_L - J_L^{min}}{\Delta J_L} + \alpha(1 - \beta(\mathbf{w}_{t+H})) \frac{V_F(\mathbf{w}_{t+H}) - J_F^{min}}{\Delta J_F} \quad (5)$$

We execute the planner in a receding-horizon fashion at a given time t , planning over a horizon of H time steps and predicting future states $\mathbf{s}_i$, $\mathbf{w}_i$ and rewards r_i^L using a zero-order model of the system dynamics and reward, as our method does not rely on gradient-based optimization. Because the time horizon is limited, we must estimate the leader’s full return, J_L , by approximating the tail rewards.

$$J_L \approx \underbrace{\sum_{i=1}^{t-1} \gamma^i r_i^L}_{\text{already incurred}} + \underbrace{\sum_{i=t}^{t+H} \gamma^i r_i^L}_{\text{within optimization horizon}} + \underbrace{\gamma^{t+H} V_L(\mathbf{s}_{t+H}, \mathbf{w}_{t+H})}_{\text{tail estimation using leader's value function}} \quad (6)$$

This limited horizon requires two practical approximations. First, because the true terminal handoff state $\mathbf{w}_T$ is unavailable at time t , we use the environment state at the end of the horizon, $\mathbf{w}_{t+H}$, as a proxy to evaluate both the follower’s value function V_F and the uncertainty penalty β . Second, using V_L as an infinite-horizon value approximation introduces a non-cooperative baseline bias: V_L assumes the leader will act purely selfishly beyond the horizon, yielding an overestimate of the true return, since the online policy may actually continue sacrificing some reward to aid the follower. Empirically, however, neither the state proxy nor the terminal value approximation adversely impacts the performance of either agent. To solve the optimization problem (5) online, PAVE employs the Cross Entropy Method (CEM) [21] over the receding horizon. A detailed listing of parameter values is provided in Appendix F.

Remark: This methodology extends to arbitrary sequences of $N > 2$ agents through recursive pairwise application. For an ordered sequence of agents $(1, \dots, N)$, each agent $i \in \{1, \dots, N-1\}$ treats agent $i+1$ as its immediate follower, adapting its policy to the successor’s value function V_{i+1} . Provided that the two-agent formulation guarantees the leader preserves the follower’s baseline performance, this framework can be applied transitively to guarantee cooperative behavior across the entire N -agent chain.

4 Experimental Analysis and Real-world Deployment

We evaluate our method in five scenarios — WAREHOUSE, WAREHOUSE PICKUP BLOCKED, WAREHOUSE PICKUP UNSEEN, MOBILE ARM, and OBSTACLE ARENA (Figure 2) — in simulation, as well as in one real-world experiment (HOUSEHOLD). For a description of each scenario, we refer the reader to Table 1. Hardware experiments are performed using the RoboMaster Platform [22]. Complete details of the simulated environments and the hardware setup are provided in Appendix C and Appendix E, respectively.

Scenario	Setting	Leader Task	Follower Task
MOBILE ARM	Heterogeneous embodiment: a robotic manipulator mounted on a mobile base.	Control the mobile base to navigate to a target object.	Control the manipulator to grasp the object; reach is restricted to a cone in front of the base.
OBSTACLE ARENA	A pusher and a navigator share a room with a bottleneck.	Push an obstacle out of the center of the room toward the nearest wall.	Traverse the bottleneck and reach the opposite wall.
WAREHOUSE	Two agents in a warehouse with three transfer areas (left, center, right).	Pick up a randomly placed object and transport it to one of the transfer areas.	Retrieve the object from the transfer area and place it at a designated shelf location.
WAREHOUSE (PICKUP-BLOCKED)	Variation in which the follower, trained on the left/right transfer areas, cannot pick up from the center.	Same as WAREHOUSE; placing at the center transfer area is infeasible for the follower.	Pick up only from the left or right transfer area; central pick-up is impossible.
WAREHOUSE (PICKUP-UNSEEN)	Variation in which the follower, trained on the left/right transfer areas, can in principle pick up from any of the three.	Same as WAREHOUSE; all three drop-offs are feasible.	Pick up from any transfer area; the central one was unseen during training.
HOUSEHOLD	Two robots operate in the same domestic environment.	Clean up dust under the table. The table has to be moved to reach it.	Traverse the narrow pathway and reach the opposite side.

Table 1: Overview of the evaluation scenarios. The WAREHOUSE family shares the same physical environment but differs in the follower’s training distribution and capability. HOUSEHOLD is a hardware-only scenario.

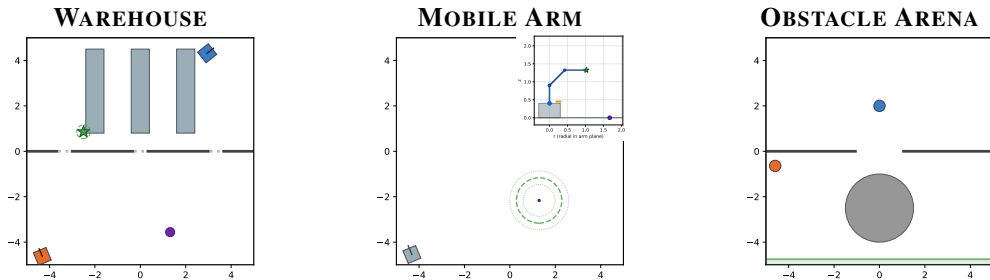


Figure 2: The three simulated scenarios. In each, a leader (first agent) reshapes the shared environment before a follower (second agent) must complete its own task from the resulting handoff state.

Baselines To the best of our knowledge, PAVE is the only method that directly addresses the problem of sequential task continuation. Cooperative Consecutive Policies (CCP) [11] train temporally consistent consecutive policies for long-horizon tasks on a single robot by modifying the current agent’s policy to maximize the value estimated by the next agent’s critic. Although CCP is structurally different (both agents share the same embodiment, as well as the state and action spaces, and are trained jointly), CCP therefore serves as inspiration for our *Finetuning* baseline. Additionally, we perform a set of ablations to analyze the contribution of each component of PAVE.

Naïve. The prior leader policy is rolled out without considering the follower’s benefit. It provides a lower bound on the performance of our proposed approach and serves as a metric for determining whether guidance is possible at all.

PAVE (w/o OOD). The proposed planner without the uncertainty filter. This serves as an ablation to assess the effectiveness of the mask in preventing the value function from being trusted in untrained regions of the state space.

PAVE (w/o π_L). The proposed planner with an uninformed prior policy. This serves as an ablation to assess the effectiveness of using the leader’s prior policy to guide the planning process.

Leader-only. The proposed planner without including the following value function in the optimization. This serves as an ablation to assess whether performance gains stem from online planning alone or from the use of the value function.

Finetuning (FT). Prior leader policy fine-tuned on the followers’ return as a terminal reward computed by a rollout of the followers’ policy from the handoff state. Because it relies on collective interaction data and further pre-deployment training (both of which PAVE forgoes), it serves as a privileged comparison.

5 Results

We evaluate our method against baselines in five simulated scenarios and one real-world deployment. Simulation results are obtained using six random seeds to compute the standard deviation. Figure 3 shows the return distributions, while Table 2 reports per-scenario success rates for both the leader and the follower, as well as their joint success.

PAVE improves follower outcomes without sacrificing leader success. We observe an average increase of 27.9 pp and a maximum increase of 48.5 pp in follower success rates across all scenarios. While the follower’s success rate in the WAREHOUSE scenario is saturated, we still observe a small improvement in the follower’s return. The leader’s success rate is generally equal to or higher than that of the NAIVE baseline. The return distributions mirror the increase in follower success rates. We observe that the follower’s return distributions are typically multimodal, with each mode corresponding to either the success or failure outcome. PAVE shifts the return distribution toward the success mode, with higher returns.

PAVE is robust to unseen handoff states. The WAREHOUSE family demonstrates the robustness of our approach to unseen handoff states. In both WAREHOUSE PICKUP BLOCKED and WAREHOUSE PICKUP UNSEEN, the follower’s value function overestimates the return of the object placed at the central gap as it interpolates between the two seen gaps. Further it wrongly extrapolates the handover value outside of transfer areas, assigning high value to areas where no handoff is possible. By employing the uncertainty filter and rejecting the misleading value estimates, PAVE is robust to this overestimation and significantly outperforms the NAIVE baseline. The PAVE (w/o OOD) ablation underperforms NAIVE and PAVE significantly, as it is led to the wrong gap or no gap at all, demonstrating that uncertainty estimation is essential for a successful handoff.

Using an OOD filter trades off residual performance with catastrophic failures. PAVE trades residual performance in familiar regions for protection against catastrophic OOD failures. The filter is deliberately conservative: while it successfully blocks over-optimistic estimates in the WAREHOUSE family, it also discards helpful smooth gradients where extrapolation would be informative. For example, in MOBILE ARM the arm only observed objects in a cone in front of it during training, but the value function still assigns higher values smoothly to the general proximity of the base. PAVE (w/o OOD) can follow this incidentally useful value estimate to the optimal handoff orientations.

Using the leader’s prior as a planning initialization never hurts and helps substantially on constrained tasks. PAVE generally outperforms PAVE (w/o π_L). MOBILE ARM and OBSTACLE ARENA benefit the most from the initial guess. Empirically, we conclude that using the leader’s prior policy as an initial guess for the optimization often improves overall performance and never impairs it.

PAVE matches finetuning without privileged information or pre-deployment training. PAVE attains performance comparable to or better than the finetuned leader in three of five scenarios and trails it in the other two. Although joint training with shared critics is typically beneficial in multi-agent settings with synchronously acting agents, the sequential, long-horizon nature of task continuation makes it difficult for reinforcement learning alone to reliably converge to the optimal solution.

Follower gains stem from value exchange and not from online planning alone. Leader-only improves leader success by up to 7.1 pp but yields no consistent follower gain, recovering at most 14.2% of PAVE’s follower improvements on the four scenarios where PAVE has headroom. Therefore, PAVE’s follower gains stem primarily from value exchange, not online planning.

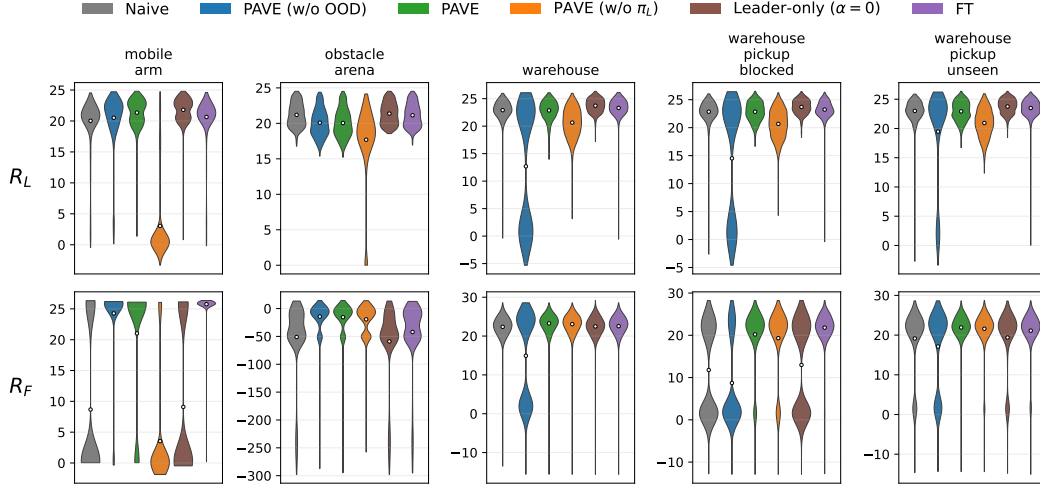


Figure 3: Distribution Plots of Leader and Follower Return under baselines and our method. Leader behavior either supports or hinders the follower, making the followers’ return distribution inherently multi-modal. Our approach successfully shifts this distribution toward the mode of follower success.

Scenario		Naive	PAVE (w/o OOD)	PAVE	PAVE (w/o π_L)	Leader-only ($\alpha = 0$)	FT
MOBILE ARM	$S_L =$	92.89 $\pm$ 5.79	95.44 $\pm$ 2.92	99.72 $\pm$ 0.36	12.89 $\pm$ 0.57	99.94 $\pm$ 0.12	95.22 $\pm$ 2.73
	$S_F =$	32.67 $\pm$ 17.84	93.83 $\pm$ 2.60	81.17 $\pm$ 5.53	13.39 $\pm$ 1.24	34.44 $\pm$ 4.78	99.50 $\pm$ 0.42
	$S_{L \wedge F} =$	30.28 $\pm$ 15.81	93.61 $\pm$ 2.81	81.00 $\pm$ 5.31	10.39 $\pm$ 0.68	34.44 $\pm$ 4.78	95.11 $\pm$ 2.73
OBSTACLE ARENA	$S_L =$	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	94.22 $\pm$ 4.72	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00
	$S_F =$	45.78 $\pm$ 11.71	84.67 $\pm$ 6.43	83.61 $\pm$ 6.64	74.17 $\pm$ 7.40	36.39 $\pm$ 11.12	54.83 $\pm$ 12.62
	$S_{L \wedge F} =$	45.78 $\pm$ 11.71	84.67 $\pm$ 6.43	83.61 $\pm$ 6.64	73.22 $\pm$ 6.72	36.39 $\pm$ 11.12	54.83 $\pm$ 12.62
WAREHOUSE	$S_L =$	99.56 $\pm$ 0.60	55.83 $\pm$ 29.20	100.00 $\pm$ 0.00	99.89 $\pm$ 0.25	100.00 $\pm$ 0.00	99.22 $\pm$ 0.92
	$S_F =$	99.33 $\pm$ 0.47	58.33 $\pm$ 26.71	99.78 $\pm$ 0.25	99.61 $\pm$ 0.36	99.72 $\pm$ 0.23	98.72 $\pm$ 1.15
	$S_{L \wedge F} =$	99.11 $\pm$ 0.83	55.67 $\pm$ 29.18	99.78 $\pm$ 0.25	99.61 $\pm$ 0.36	99.72 $\pm$ 0.23	98.50 $\pm$ 1.27
WAREHOUSE PICKUP BLOCKED	$S_L =$	99.28 $\pm$ 1.08	62.56 $\pm$ 14.58	100.00 $\pm$ 0.00	99.94 $\pm$ 0.12	100.00 $\pm$ 0.00	99.72 $\pm$ 0.30
	$S_F =$	49.78 $\pm$ 10.26	32.50 $\pm$ 7.01	89.94 $\pm$ 10.02	85.67 $\pm$ 1.53	55.50 $\pm$ 7.65	98.11 $\pm$ 2.02
	$S_{L \wedge F} =$	49.44 $\pm$ 10.21	31.50 $\pm$ 7.37	89.94 $\pm$ 10.02	85.67 $\pm$ 1.53	55.50 $\pm$ 7.65	98.00 $\pm$ 1.95
WAREHOUSE PICKUP UNSEEN	$S_L =$	99.89 $\pm$ 0.25	83.06 $\pm$ 10.13	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	99.89 $\pm$ 0.16
	$S_F =$	85.44 $\pm$ 5.03	73.89 $\pm$ 13.87	98.22 $\pm$ 0.76	96.56 $\pm$ 1.42	86.61 $\pm$ 5.54	95.00 $\pm$ 1.66
	$S_{L \wedge F} =$	85.44 $\pm$ 5.03	73.67 $\pm$ 14.02	98.22 $\pm$ 0.76	96.56 $\pm$ 1.42	86.61 $\pm$ 5.54	95.00 $\pm$ 1.66
HOUSEHOLD (on hardware)	$S_L =$	100.00	—	100.00	—	—	—
	$S_F =$	30.00	—	70.00	—	—	—
	$S_{L \wedge F} =$	30.00	—	70.00	—	—	—

Table 2: Leader / follower / joint success rates (%) per scenario across modes, each as mean $\pm$ std across 6 seeds. S_L / S_F = leader/follower success rate. $S_L \wedge S_F$ is computed per seed (counting the rollouts in which both leader and follower succeeded) and then aggregated across seeds. **Bold** marks the maximum between methods without access to privileged information. FT serves as a privileged comparison; **bold** when it outperforms PAVE. HOUSEHOLD is hardware-only and reports the success rates of 20 leader-follower deployments with randomized initial conditions.

PAVE transfers from simulation to a physical household setup. In 20 deployments with randomized initial conditions, we compare the performance of the NAIVE baseline with our PAVE method, running online in the HOUSEHOLD scenario (Figure 1). With both NAIVE and PAVE, the leader succeeds in every instance; however, PAVE increases the follower’s success rate by 40 pp. The real collision dynamics differ substantially from those in simulation (which assume no friction and simple velocity matching), highlighting the strength of online replanning in compensating for prediction error.

6 Conclusion

Extrapolating from current developments, robots will soon be ubiquitous in daily life. While current research often focuses on building optimal systems for individual tasks, the growing number

of robots in shared spaces will require simple and efficient methods and protocols to coordinate heterogeneous robotic systems. In this paper, we took a step towards such an interface and proposed the use of the environmental value function to serve as such a protocol. We have shown that an *agent’s value function*, when paired with an *uncertainty filter*, serves as an effective interface between learning-based agents. We demonstrate that our method improves follower success rates by 27.9 pp on average and by up to 48.5 pp, with the greatest gains occurring when the leader’s selfish choices are most detrimental to the follower’s success. This suggests that the value-function interface is most effective when cooperation matters most. Uncertainty filtering through out-of-distribution detection is necessary to prevent catastrophic failures. Without it, the leader may follow the follower’s value function into unseen environmental states, which degrades performance.

Limitations and Future Work This work focuses on *two agents acting sequentially in a pre-defined order*. One axis of future research lies in the number of possible followers. If the sequence of agents acting in the environment is either unknown or not predetermined, one leader could adapt to the needs of multiple followers; a possible solution is optimizing a weighted combination of their value functions or lexicographically optimizing in an order of priority. A critical direction for future work is to tackle nested optimization as a second axis of scaling. By extending our planning horizon beyond the immediate follower, future models can optimize for complex, multi-step sequences involving multiple downstream agents. Due to runtime constraints of online adaptation, the planning horizon is limited. As a result, the handoff state has to be approximated with the state prediction at the end of the planning horizon. This issue is partly alleviated by the uncertainty filter, since mid-leader-episode handoff states are often not observed by the follower, reducing the influence of intermediate approximations until the end of the episode approaches. Nonetheless, this remains a critical challenge when considering environment modifications that require a longer acting horizon. This work assumes that both agents “*share a common representation of the environment $\mathbf{w}_T$* ”. Because robots often serve different purposes, their world representations may also be task-specific. Future work on either transformations between different representations or a general shared world representation could help extend this framework. Furthermore, we assume that the follower shares a truthful value function and uncertainty estimate with the leader. A malicious follower could mislead the leader to produce unfavorable or even dangerous environmental states, making further research into approaches to increase safety necessary.

Acknowledgments

This work is supported by European Research Council (ERC) Project 949940 (gAIA).

References

- [1] H. Hu, A. Lerer, A. Peysakhovich, and J. Foerster. “Other-Play” for Zero-Shot Coordination. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4399–4410. PMLR, Nov. 2020. URL <https://proceedings.mlr.press/v119/hu20a.html>.
- [2] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1504–1509, July 2010. ISSN 2374-3468, 2159-5399. doi:10.1609/aaai.v24i1.7529. URL <https://ojs.aaai.org/index.php/AAAI/article/view/7529>.
- [3] S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, 242:132–171, Jan. 2017. ISSN 00043702. doi:10.1016/j.artint.2016.10.005. URL <https://linkinghub.elsevier.com/retrieve/pii/S0004370216301266>.
- [4] J. G. Ribeiro, G. Rodrigues, A. Sardinha, and F. S. Melo. TEAMSTER: Model-based reinforcement learning for ad hoc teamwork. *Artificial Intelligence*, 324:104013, Nov. 2023. ISSN 0004-3702. doi:10.1016/j.artint.2023.104013. URL <https://www.sciencedirect.com/science/article/pii/S0004370223001595>.
- [5] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfef4-Abstract.html.
- [6] Q. Li, F. Gama, A. Ribeiro, and A. Prorok. Graph Neural Networks for Decentralized Multi-Robot Path Planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11785–11792, Las Vegas, NV, USA, Oct. 2020. IEEE. ISBN 978-1-7281-6212-6. doi:10.1109/IROS45743.2020.9341668. URL <https://ieeexplore.ieee.org/document/9341668/>.
- [7] C. Yu, A. Velu, E. Vinitzky, 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, Dec. 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html.
- [8] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html.
- [9] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. ISSN 2374-3468, 2159-5399. doi:10.1609/aaai.v32i1.11794. URL <https://ojs.aaai.org/index.php/AAAI/article/view/11794>.
- [10] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4295–4304. PMLR, July 2018. URL <https://proceedings.mlr.press/v80/rashid18a.html>.
- [11] J. Erskine and C. Lehnert. Developing Cooperative Policies for Multi-Stage Reinforcement Learning Tasks. *IEEE Robotics and Automation Letters*, 7(3):6590–6597, July 2022. ISSN 2377-3766. doi:10.1109/LRA.2022.3174258. URL <https://ieeexplore.ieee.org/abstract/document/9772966>.

- [12] R. Dhakal, D. M. Nguyen, T. Silver, X. Xiao, and G. J. Stein. Anticipatory task and motion planning: Improved rearrangement in persistent continuous-space environments. *IEEE Robotics and Automation Letters*, 11(2):1850–1857, 2026. doi:10.1109/LRA.2025.3631357.
- [13] O. Holland and C. Melhuish. Stigmergy, Self-Organization, and Sorting in Collective Robotics. *Artificial Life*, 5(2):173–202, Apr. 1999. ISSN 1064-5462, 1530-9185. doi:10.1162/106454699568737. URL <https://direct.mit.edu/artl/article/5/2/173-202/2315>.
- [14] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, June 2000. ISSN 0167739X. doi:10.1016/S0167-739X(00)00042-X. URL <https://linkinghub.elsevier.com/retrieve/pii/S0167739X0000042X>.
- [15] O. Rappel, M. Amir, and A. M. Bruckstein. Stigmergy-based, dual-layer coverage of unknown regions. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 1439–1447, 2023.
- [16] Z. Gao and A. Prorok. Constrained Environment Optimization for Prioritized Multi-Agent Navigation. *IEEE Open Journal of Control Systems*, 2:337–355, 2023. ISSN 2694-085X. doi:10.1109/OJCSYS.2023.3316090. URL <https://ieeexplore.ieee.org/document/10251921>.
- [17] Z. Gao, G. Yang, and A. Prorok. Co-Optimizing Reconfigurable Environments and Policies for Decentralized Multiagent Navigation. *IEEE Transactions on Robotics*, 41:4741–4760, 2025. ISSN 1941-0468. doi:10.1109/TRO.2025.3588449. URL <https://ieeexplore.ieee.org/document/11077995>.
- [18] H. X. Li, M. Amir, and A. Prorok. Scaling Multi-Agent Environment Co-Design with Diffusion Models, Nov. 2025. URL <http://arxiv.org/abs/2511.03100>. arXiv:2511.03100.
- [19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, Aug. 2017. URL <http://arxiv.org/abs/1707.06347>. arXiv:1707.06347 [cs.LG].
- [20] Y. Sun, Y. Ming, X. Zhu, and Y. Li. Out-of-Distribution Detection with Deep Nearest Neighbors. In *Proceedings of the 39th International Conference on Machine Learning*, pages 20827–20840. PMLR, June 2022. URL <https://proceedings.mlr.press/v162/sun22d.html>.
- [21] R. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology And Computing In Applied Probability*, 1(2):127–190, Sept. 1999. ISSN 1573-7713. doi:10.1023/A:1010091220143. URL <https://doi.org/10.1023/A:1010091220143>.
- [22] J. Blumenkamp, A. Shankar, M. Bettini, J. Bird, and A. Prorok. The Cambridge RoboMaster: An agile multi-robot research platform. In *International Symposium on Distributed Autonomous Robotic Systems*, pages 439–456. Springer, 2024.
- [23] R. Mirsky, I. Carlucho, A. Rahman, E. Fosong, W. Macke, M. Sridharan, P. Stone, and S. V. Albrecht. A Survey of Ad Hoc Teamwork Research. In D. Baumeister and J. Rothe, editors, *Multi-Agent Systems*, pages 275–293, Cham, 2022. Springer International Publishing. ISBN 978-3-031-20614-6. doi:10.1007/978-3-031-20614-6_16.
- [24] C. S. De Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. Torr, M. Sun, and S. Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.
- [25] J. Jiang and Z. Lu. I2Q: A Fully Decentralized Q-Learning Algorithm. *Advances in Neural Information Processing Systems*, 35:20469–20481, Dec. 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8078e8c3055303a884ffae2d3ea00338-Abstract-Conference.html.
- [26] Y. Zang, J. He, K. Li, H. Fu, Q. Fu, and J. Xing. Sequential cooperative multi-agent reinforcement learning. In *Proceedings of the 2023 international conference on autonomous agents and multiagent systems*, pages 485–493, 2023.

- [27] G. Theraulaz and E. Bonabeau. A Brief History of Stigmergy. *Artificial Life*, 5(2):97–116, Apr. 1999. ISSN 1064-5462. doi:10.1162/106454699568700. URL <https://doi.org/10.1162/106454699568700>.
- [28] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, Nov. 2006. ISSN 1556-6048. doi:10.1109/MCI.2006.329691. URL <https://ieeexplore.ieee.org/document/4129846>.
- [29] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.
- [30] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.
- [31] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius. Sample-efficient cross-entropy method for real-time planning. In J. Kober, F. Ramos, and C. Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1049–1065. PMLR, 16–18 Nov 2021. URL <https://proceedings.mlr.press/v155/pinneri21a.html>.
- [32] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Byey7n05FQ>.
- [33] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. In *ICML*, 2022.
- [34] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.
- [35] J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 132(12):5635–5662, 2024.
- [36] Q. Li, W. Lin, Z. Liu, and A. Prorok. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540, 2021.
- [37] A. Shankar, S. Elbaum, and C. Detweiler. Freyja: A full multirotor system for agile & precise outdoor flights. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 217–223. IEEE, 2021.

A Symbols

Symbol	Description
<i>Agents & Policies</i>	
π_L	Leader’s policy
π_F	Follower’s policy (fixed, independently trained)
r_L, r_F	Leader’s and follower’s reward functions
<i>States, Distributions, and Dynamics</i>	
$\mathbf{w}$	Shared environment state
$\mathbf{s}^L, \mathbf{s}^F$	Leader’s and follower’s private states
$\mathbf{w}_t$	Environment state at time t
$\mathbf{s}_t^L, \mathbf{s}_t^F$	Leader’s and follower’s private states at time t
$\mathbf{w}_T$	Terminal environment state (handoff state from leader to follower)
$\mathbf{w}_{t+H}$	Environment state at the end of the planning horizon
ρ_L, ρ_F	Initial distributions of leader’s and follower’s private states
ρ_W	Initial distribution of the environment state
$H_T(\pi_L)$	Handoff distribution induced by leader policy π_L over T steps
T	Leader’s episode length
γ	Discount factor
<i>Returns and Value Functions</i>	
$J_L(\pi_L)$	Leader’s expected return
$J_F(\pi_F, \mathbf{w})$	Follower’s expected return from handoff state $\mathbf{w}$
$V_F(\mathbf{w})$	Environmental value function of the follower
$V_F^{\text{critic}}(\mathbf{s}, \mathbf{w})$	PPO-trained follower critic (depends on private state)
$V_L^{\text{critic}}(\mathbf{s}, \mathbf{w})$	Leader’s value function (used for tail estimation in planning)
α	Cooperation weighting factor
$J_L^{\min}, \Delta J_L$	Leader return normalization constants
$J_F^{\min}, \Delta J_F$	Follower value normalization constants
<i>OOD Uncertainty Filter</i>	
$\beta(\mathbf{w}_T)$	KNN-based OOD score, $\in [0, 1]$
K	Number of nearest neighbors
$\text{neighbor}(\mathbf{w}_T, k)$	k -th nearest neighbor of $\mathbf{w}_T$ in the training distribution
τ	Distance threshold below which states are considered in-distribution
$d_{\max}$	Distance ceiling above which states are fully out-of-distribution
<i>Planning</i>	
H	Planning horizon (receding-horizon)
r_i^L	Leader’s reward at time i
$\mathbf{s}_i, \mathbf{w}_i$	Predicted states within the planning horizon

B Extended Related Work

Ad Hoc Teamwork Stone et al. [2] define the challenge of creating ad hoc teams of autonomous agents as: *“To create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members.”* PLASTIC [3] addresses the problem in two variants. PLASTIC-Model learns teammate models from prior teams and plans against them at deployment, while PLASTIC-Policy learns a library of cooperative policies and selects the best-fitting one online based on the observed teammate behavior. TEAMSTER [4] builds on ideas in PLASTIC and not only learns a model of the teammates, but also a model of the environment. For a detailed survey of Ad Hoc Teamwork, the reader is referred to [23]. Our approach differs from Ad Hoc Teamwork in that it explicitly allows direct communication between agents, whereas the Ad Hoc Teamwork literature typically omits such communication. We further consider agents that perform distinct tasks rather than receive a shared payoff.

Zero Shot Coordination Zero Shot Coordination makes different assumptions than Ad Hoc Teamwork, as it considers games where players must choose and fix their strategies prior to being deployed together. Unlike Ad Hoc Teamwork, this does not allow for online adaptation to teammates, but requires fixed policies that are robust to a range of different partners. Other-Play [1] proposes a training framework that trains policies to be robust to independently trained agents using equivalence mappings, thereby preventing task-specific conventions arising from Self-Play. As PAVE specifically assumes message-passing and adaptation to the previously unseen partner, it is not a Zero-Shot Coordination method, but addresses a related problem.

Multi-Agent Reinforcement Learning and Shared Critics Multi-Agent Reinforcement Learning extends classical Single-Agent Reinforcement Learning to Decentralized POMDPs in which multiple agents jointly act in the same environment. MARL algorithms are typically classified by whether they are centralized (C) or decentralized (D) at training (T) and execution (E) time. The CTCE paradigm considers all actors as a single agent employing single-agent RL on their joint state and action space. These approaches typically do not scale well with the number of agents. Approaches like IPPO [24] and I2Q [25] belong to the DTDE paradigm and consider each agent as an individual learner interacting with the environment at the same time. While suffering less from poor scaling in the number of agents, multiple agents learning in the same environment introduce non-stationarity in the environmental dynamics from the perspective of any single agent. CTDE approaches like MAPPO [7], MADDPG [8], COMA [9], and QMIX [10] introduce a shared critic function to the training process, targeting non-stationary and credit assignment problems to stabilize the training process.

MARL typically does not consider long-horizon sequential tasks in which one agent acts for an entire episode before handing off the environment to a second agent. In MARL, sequentiality typically refers to the sequential execution of a single action at one timestep and appears in the context of credit assignment. Zang et al. [26] propose a sequential MARL framework in which the value function is factorized according to the sequence of agents acting on the environment. Erskine et al. [11] consider sequential tasks with handoff, but restrict their setting to multiple policies acting on the same robot. Because the embodiment remains unchanged, achieving consistency between different agents is ensured by training the agents with a mixture of subsequent Q-functions. In contrast, our approach considers agents with different embodiments, distinct action spaces, and modifiable environments.

Stigmergy Stigmergy is a natural process that organizes the behavior of animal swarms [14, 27, 13]. Ants, for example, leave pheromone trails while foraging, which other ants can follow to a food source. Over time, the paths that individuals take provably converge to optimal solutions, which is why stigmergy has inspired the class of Ant Colony Optimization (ACO) algorithms [28]. Although our proposed approach also considers modifications to the environment to influence other agents’ behavior, it directly optimizes the follower’s return instead of using the environment as an

implicit communication variable. Therefore, while our approach shares some of the characteristics of stigmergy, it is not strictly stigmergic.

Environmental Co-Optimization As we consider sequential scenarios where the agents cooperate solely through a shared environment, our work connects to prior work on environmental optimization. Gao et al. [16, 17], and Li et al. [18] consider the environment as a decision variable together with the policies of agents acting within it and jointly optimize both in a bi-level iterative optimization scheme. While our work also considers the effects of a given environment on the return and task success of a given agent, we clearly differentiate from existing literature in two ways. Firstly, we do not consider the environment as an explicit decision variable, as it can only be modified indirectly through the leading agent’s behavior, and secondly, we assume that the second agent’s policy does not adapt to the changing environment.

Sampling-Based Model Predictive Control with Learned Cost Functions PAVE uses a sampling-based MPC formulation [29] with the Cross-Entropy Method [21] as the inner optimizer, similar to its use in model-based RL planners such as PETS [30]; recent CEM variants such as iCEM [31] are complementary and could be incorporated. For tail estimation, PAVE bootstraps the leader’s return using its own value function V_L , following standard practice in value-augmented MPC [32, 33]. The general principle of pairing a search procedure with a learned value estimate was demonstrated at scale by AlphaGo Zero [34], which used a learned value function to evaluate leaf nodes within Monte Carlo Tree Search. The role of the follower’s value function V_F differs as it serves as the coordination interface through which the two independently trained agents communicate their preferences. PAVE additionally employs a k -nearest-neighbor distance-based Out-Of-Distribution detector [20, 35] to down-weight the value-function term in environment state regions in which the follower critic has not been trained to prevent inaccurate value approximations.

Learned Communication Multi-agent learning has been shown to benefit from learned communication channels [5, 6, 36], where messages emerge from joint training and condition agents’ policies on one another at every timestep. PAVE differs in two ways. First, communication is one-shot: the follower transmits its value function once, and the leader plans and acts without further exchange. Second, the transmitted message has a known meaning (the follower’s expected return from a given environment state), instead of being a task- and training-team-specific representation, only interpretable between co-trained agents. PAVE’s message is learned but interpretable, enabling pairing of arbitrary agents that have a shareable value function.

C Simulation Environments

All simulation environments are custom and written in numpy. The following explains the rewards structure of individual environments and lists their numerical values. Figure 2 depicts OBSTACLE ARENA, MOBILE ARM and WAREHOUSE. Figure 1 and Figure 4 show the HOUSEHOLD scenario.

C.1 MOBILE ARM

The leader must steer the mobile base to a specified distance from the target object and then terminate the episode by outputting a termination signal from its action space. The follower controls a 4-DoF arm, consisting of the arm’s yaw and three parallel joints to pick up the object and place it in a tray on the mobile base. The object is picked up and placed when the end effector comes sufficiently close to the corresponding pick-up and placement points.

Table 3: MOBILE_ARM environment parameters.

Leader		Follower	
Parameter	Value	Parameter	Value
step_penalty	0.01	step_penalty	0.01
progress_weight	1	progress_weight	1
goal_reward	25	pickup_reward	1
gaussian_sigma	0.3	goal_reward	25
		collision_penalty	0

C.2 OBSTACLE ARENA

The leader has to clear the lower half of the arena by pushing the obstacle out of the center and into contact with any wall; a follower must then traverse the room by navigating through a narrow pathway and exiting it at the lower boundary. Their objectives conflict when the leader pushes the obstacle into contact with the central wall, blocking the passageway.

Table 4: OBSTACLE_ARENA environment parameters.

Leader		Follower	
Parameter	Value	Parameter	Value
goal_reward	25	goal_reward	25
		collision_penalty	300
		progress_weight	1
		distance_weight	0.1

C.3 WAREHOUSE Family

PAVE is evaluated on three variations of the WAREHOUSE with the same underlying reward structure. The leader is tasked with picking up an object in the lower half of the arena and transporting it to one of the three transfer areas. The follower has to navigate through the shelves to the transfer area, pick up the object, and place it at a goal position on the shelf. Pickup and placing occur automatically when the leader or follower is close enough to the object and oriented towards it.

Table 5: WAREHOUSE environment parameters.

Leader		Follower	
Parameter	Value	Parameter	Value
step_penalty	0.01	step_penalty	0.01
step_penalty_carrying	0.05	step_penalty_carrying	0.1
pickup_reward	1	pickup_reward	4
goal_reward	25	goal_reward	25
progress_weight	1	progress_weight	1
		collision_penalty	0.1

C.4 HOUSEHOLD

This scenario is similar to OBSTACLE ARENA and only differs in the leader’s objective. Instead of pushing the obstacle into contact with the wall, it has to move the obstacle out of the way to reach the dirt that is positioned under the table.

Table 6: HOUSEHOLD environment parameters.

Leader		Follower	
Parameter	Value	Parameter	Value
step_penalty	0.01	progress_weight	3
progress_weight	3	goal_reward	25
wall_collision_penalty	0.1	collision_penalty	10
pickup_reward	25	distance_weight	0.1

D Training and Model Selection

All leaders and followers across all scenarios use policy networks with two hidden layers of size 64 each, with a Gaussian policy head (SB3 defaults). Table 8 shows the training budget for the leader and follower per scenario. The training steps were increased until policy convergence was reached.

Table 2 was generated by evaluating our method and the baseline over six independent training and evaluation seeds. PAVE assumes capable actors, so we discard policies with under 70% individual success. Applied uniformly across all baselines, this ensures a fair cross-method comparison. This removes noise from followers that fail regardless of interaction method while preserving variance among competent policies. No further selection of seeds was performed based on the performance of our method or baselines.

Table 7: PPO (Stable-Baselines3, MlpPolicy) algorithm hyperparameters.

Hyperparameter	Value
Learning rate	0.0003
Rollout length n_{steps} (per env)	2048
Minibatch size	64
Epochs per update	10
Discount γ	0.99
GAE λ	0.95
PPO clip range ϵ	0.2
Entropy coef. c_{ent}	0
Value-function coef. c_{vf}	0.5
Max gradient norm	0.5

Table 8: Per-agent training budget: Follower/leader total environment steps

Scenario	Follower steps	Leader steps
MOBILE_ARM	2 M	1 M
OBSTACLE_ARENA	500 k	400 k
WAREHOUSE	1.5 M	500 k
WAREHOUSE_PICKUP_BLOCKED	1.5 M	500 k
WAREHOUSE_PICKUP_UNSEEN	1.5 M	500 k
HOUSEHOLD	1.5 M	400 k

All `Finetuning` experiments were run for an additional 1.5M steps. The return signal is computed as the sum of the normalized leader return and the normalized follower return, with the latter scaled by $\alpha = 1$. This differs from the $\alpha = 4$ used in `PAVE`. We made this choice because `Finetuning` with $\alpha = 1$ yielded higher success rates than with $\alpha = 4$, so we report only the best-performing version of the `Finetuning` baseline.

E Hardware Setup

For the deployment of the HOUSEHOLD scenario, we used two RoboMaster [22] robots. The robots were velocity-controlled using the Freyja controller [37] at 50 Hz. Both the robots and the box representing the table in the scenario were tracked using OptiTrack, and the resulting pose data were used as feedback for the policies at 50 Hz. The dirt was purely virtual and initialized at the table’s initial position. Policies were trained in simulation and deployed on the real robots without further modification. PAVE used the nominal model from simulation to optimize trajectories and adapt the leader’s policy. Figure 4 shows the experimental setup.

Trials To compare Naive with PAVE, both methods were deployed across 20 trials with varied initial conditions; each trial used the same starting condition for Naive and PAVE so that the methods could be compared on identical configurations. Between trials, the initial conditions were randomized. The table was placed in the center of the lower half of the arena at different orientations, and the leader and follower were placed in the lower and upper halves of the arena, respectively.

Sim-To-Real Transfer and Failure Modes In the absence of collisions, RoboMaster robots can be velocity-controlled with minimal tracking error, which is why the sim-to-real gap for the follower was negligible. In simulation, the interaction between the leader and the table is modeled using simple velocity matching, where the leader effectively imposes a translational and rotational velocity on the table depending on the contact angle. This resulted in a mismatch between simulation and real-world deployment, as the friction between the floor and the box is relatively high, but it did not necessarily diminish performance, since receding-horizon replanning could compensate for the model error. Nevertheless, real-world trajectories tended to be less smooth: the leader frequently required several push attempts within a single trial to successfully reposition the object, re-engaging when the initial push slipped or under-rotated.

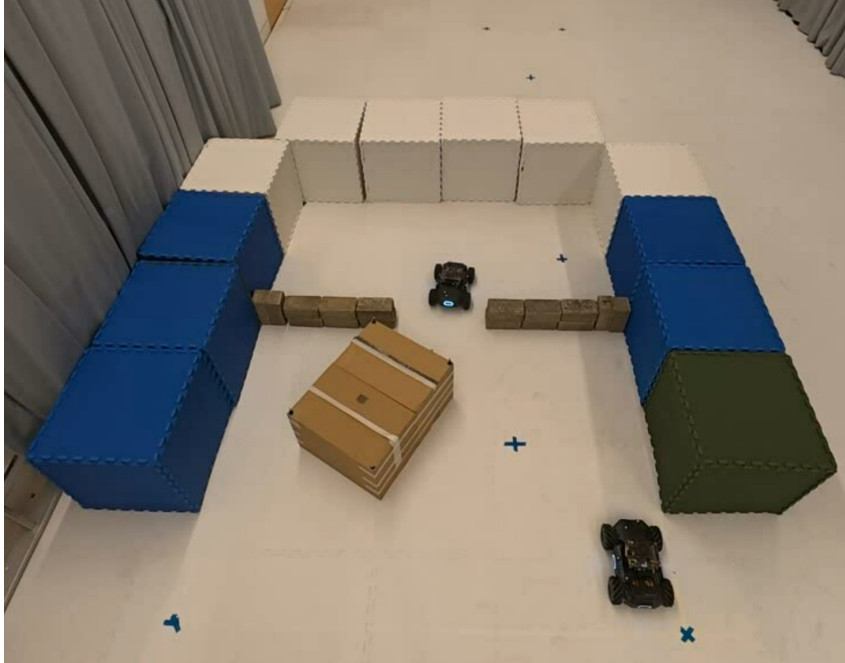


Figure 4: Hardware setup: Two RoboMaster robots and the table (box) in the HOUSEHOLD arena, tracked by OptiTrack. The dirt is virtual and co-located with the initial table position.

F Planner Details

The CEM planner uses a Gaussian action distribution with elite selection. We warm-start the action distributions by setting the mean to the actions obtained from rolling out the naive leader policy through the dynamics, and we initialize the covariance to σ_{noise} . At each iteration, we draw samples from the action distribution and evaluate them with respect to the objective defined in Equation (5). We then select the best samples and update the action distribution by fitting a Gaussian to them. To prevent covariance collapse, we impose a lower bound on the covariance.

We instantiate the cost using the same parameters and procedures across all scenarios. Specifically, we use a follower value weighting term of $\alpha = 4$, a discount factor of $\gamma = 0.99$, and normalize the leader and follower returns to $[0, 1]$ using the returns from 200 rollout episodes of the prior policies.

Table 9: Cross-entropy-method (CEM) planner hyperparameters per scenario.

Scenario	Horizon	Samples	Iters	Elite frac.	σ_{noise}	Min. std	Replan	Dist.
MOBILE_ARM	30	200	4	0.2	0.5	0.1	5	gaussian
OBSTACLE_ARENA	40	300	10	0.3	0.5	0.208	3	gaussian
WAREHOUSE	40	500	10	0.2	1.5	0.1	5	gaussian
WAREHOUSE_PICKUP_BLOCKED	40	500	10	0.2	1.5	0.1	5	gaussian
WAREHOUSE_PICKUP_UNSEEN	40	500	10	0.2	1.5	0.1	5	gaussian
HOUSEHOLD	20	100	5	0.3	0.5	0.2	5	gaussian

G Case Study: Value Function and Uncertainty Estimate

Figure 5 shows the trajectories of the leader and follower policies for the PAVE (w/o OOD) baseline and PAVE in the WAREHOUSE PICKUP BLOCKED scenario, along with the value function, and uncertainty-weighted value function evaluated over the transfer area. This example demonstrates the effectiveness of combining a value function with an uncertainty metric. The value function catastrophically overestimates the return of the central transfer area, even though pick-up is not possible from this location. Without an uncertainty estimate, PAVE (w/o OOD) is unaware that the follower has not been trained on experience involving the central drop-off area, and therefore trusts the estimate and drops the object in the center, leading to an adverse outcome for the follower. In contrast, PAVE optimizes the uncertainty-weighted value function, disregarding the erroneous value estimate and instead selecting the next-best transfer area with low uncertainty.

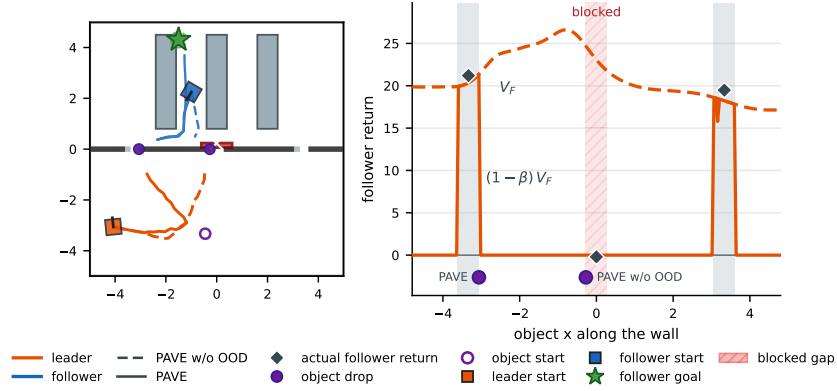


Figure 5: Comparison of trajectories PAVE (w/o OOD) (dashed) and PAVE (solid) in one example of the WAREHOUSE PICKUP BLOCKED scenario. **Left:** Leader/Follower Trajectories. **Right:** Value function and uncertainty-weighted value function evaluated along the middle of the arena. Areas in which a drop is possible are marked in grey, blocked areas are marked in red.

H Decomposition of the Leader Objective

In this section, we call a leader policy *selfish* if it optimizes only the leader’s individual objective, without regard to the follower. Formally, given a feasible leader policy class Π_L , a selfish leader policy is any

$$\pi_L^\circ \in \arg \max_{\pi_L \in \Pi_L} J_L(\pi_L).$$

For a cooperation weight $\alpha \geq 0$, we define the cooperative objective

$$U_\alpha(\pi_L) = J_L(\pi_L) + \alpha \mathbb{E}_{\mathbf{w} \sim H_T(\pi_L)}[J_F(\pi_F, \mathbf{w})].$$

We call a leader policy U_α -optimal if it maximizes U_α over Π_L . For any alternative leader policy π'_L , the improvement over the selfish leader decomposes as

$$U_\alpha(\pi'_L) - U_\alpha(\pi_L^\circ) = \alpha (\mathbb{E}_{\mathbf{w} \sim H_T(\pi'_L)}[J_F(\pi_F, \mathbf{w})] - \mathbb{E}_{\mathbf{w} \sim H_T(\pi_L^\circ)}[J_F(\pi_F, \mathbf{w})]) - [J_L(\pi_L^\circ) - J_L(\pi'_L)].$$

Consequently, for $\alpha > 0$, adapting the leader can improve the cooperative objective exactly when the induced follower gain exceeds the leader loss scaled by $1/\alpha$:

$$\mathbb{E}_{\mathbf{w} \sim H_T(\pi'_L)}[J_F(\pi_F, \mathbf{w})] - \mathbb{E}_{\mathbf{w} \sim H_T(\pi_L^\circ)}[J_F(\pi_F, \mathbf{w})] > \frac{J_L(\pi_L^\circ) - J_L(\pi'_L)}{\alpha}.$$

For $\alpha = 0$, the cooperative objective reduces to the selfish leader objective, so any selfish leader’s policy is already optimal.

This identity also clarifies when no adaptation can improve over the selfish leader. If the selfish leader already produces a handoff state that is optimal for the follower, i.e.,

$$\mathbb{E}_{\mathbf{w} \sim H_T(\pi_L^\circ)}[J_F(\pi_F, \mathbf{w})] = \sup_{\pi_L \in \Pi_L} \mathbb{E}_{\mathbf{w} \sim H_T(\pi_L)}[J_F(\pi_F, \mathbf{w})],$$

then no feasible deviation can improve the follower term, while every deviation does not increase the leader term. In that case, the selfish leader is already U_α -optimal for every $\alpha \geq 0$.

I Reproducibility Statement

Our method is described in Section 3, with architectures, hyperparameters, training details, and OOD parameters in the Appendix. The complete code is publicly available at <https://github.com/proroklab/PAVE>.