ARMS: Automatic Reward Shaping for Sparse-Reward Multi-Agent Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Sparse rewards are a major bottleneck in multi-agent reinforcement learning (MARL), where simultaneous learning induces non-stationarity and makes reward design especially delicate. Reward shaping can accelerate learning, but in the multi-agent setting it must preserve the strategic structure of the problem rather than merely improve short-term optimization. We propose **A**utomatic **R**eward-shaping in **M**ulti-agent **S**ystems (**ARMS**), a self-supervised reward shaping framework for MARL that learns dense shaping signals from sparse environmental rewards through trajectory ranking. Since single-agent trajectory-ranking guarantees do not directly transfer to MARL, we reformulate policy invariance through conditional best-response reasoning, and show that if certain conditions hold, then using shaping rewards preserves each agent's best-response set under fixed opponent policies, and consequently preserve the set of Nash equilibria. Guided by this perspective, ARMS alternates between policy learning and reward learning while sharing shaping parameters across agents for effciency. Experiments in a partially observable multi-agent pathfinding domain show that ARMS improves sampling efficiency under increasing reward sparsity and agent count, generalizes to unseen environments, and reveals a MARL-specific failure mode in which limited exploration and coupled policy–reward dynamics induce oscillatory behavior. Increasing exploration mitigates this effect and stabilizes learning. To the best of our knowledge, ARMS is the first automatic reward shaping framework for MARL whose design is motivated by a game-theoretic equilibrium-preservation result.

1 Introduction

Multi-Agent Reinforcement Learning (MARL) has seen substantial progress, which is evident from the increasing number of papers and surveys in the field in recent years (Sunehag et al., 2018; Rashid et al., 2020; Nguyen et al., 2018; Canese et al., 2021; Wong et al., 2023; Zhu et al., 2024; Oroojlooy & Hajinezhad, 2023). A central challenge in this setting is learning effectively from sparse rewards.

Sparse rewards are significantly more problematic in MARL due to non-stationarity and co-adaptation between agents, an element not present in single agent reinforcement learning (SARL). A common way to address sparse rewards is *reward shaping*, namely augmenting the training signal with additional rewards that guide learning toward desirable behavior, or replacing the reward completely. Existing reward shaping methods are typically developed for single-agent settings, and extending them to MARL is non-trivial because the shaping signal must remain meaningful under coupled learning dynamics without altering the underlying strategic structure of the problem. In particular, reward shaping is risky because it may alter the set of optimal policies in SARL, or, in the multi-agent setting, the set of Nash equilibria (Nash, 1951). For this reason, reward shaping methods that preserve Nash equilibria are especially attractive.

In reinforcement learning (RL), sparse reward functions are often easier to specify than dense ones, but they provide weak supervision and can substantially slow learning. Reward shaping aims to transform this sparse feedback into a denser training signal without changing the underlying solution of the problem. In MARL, this is especially delicate because multiple agents learn and act simultaneously in a shared environment, continually adapting their policies to one another as learning progresses. This gives rise to the inherent

MARL challenge of non-stationarity (Albrecht et al., 2024, Section 5.4.1), under which the environment appears non-stationary from the perspective of each individual agent. Moreover, unlike SARL where the relevant solution is an optimal policy, in MARL the relevant solution is defined through mutual best responses and Nash equilibria: no agent can increase its expected return[1] by unilaterally altering its policy.

Designing such an informative dense reward is challenging even in single-agent settings (Abbeel & Ng, 2004; Sutton & Barto, 2018). In multi-agent environments, this challenge becomes even harder, since the reward should encourage coordinated behavior between agents while preserving the set of Nash equilibria of the underlying problem (Oroojlooy & Hajinezhad, 2023). This is precisely why reward shaping is both appealing and riskier in MARL: it has the potential to accelerate learning, but should not induce a shaping signal which changes the agents' objectives *across all agents*.

Traditionally, *reward shaping* refers to the hand-design of an additional reward signal by an expert, using domain knowledge or intuition to guide learning toward optimal policies. See Ng et al. (1999) and Lu et al. (2011) for more information.

However, a more recent framework, called Self-Supervised Online Reward Shaping (SORS) (Memarian et al., 2021), which removes the need for hand-designing reward was developed for SARL. The main idea is to use a parameterized method to infer a dense reward function by ranking trajectories based on the environment's sparse reward. Roughly speaking, the paper proposes a framework for training a parameterized reward function and shows theoretically that any two reward functions which similarly rank every pair of trajectories also induce the same set of optimal policies - which is the main motivation for inferring a reward function based on trajectory ranking.

In this paper, we propose **A**utomatic **R**eward-shaping in **M**ulti-agent **S**ystems (**ARMS**), a framework for automatic reward shaping in MARL (Algorithm 1, Section 5). ARMS is built around a multi-agent view of trajectory-based reward learning, in which policy invariance is formulated through conditional best-response reasoning rather than global optimality (Section 4). The framework alternates between reinforcement learning and reward-learning steps, using sparse environmental rewards to supervise trajectory ranking. We further examine how multi-agent non-stationarity affects reward learning and training stability (Section 6.4 and Section 6.3). To the best of our knowledge, ARMS is the first automatic reward shaping framework for MARL whose design is motivated by a game-theoretic equilibrium-preservation result.

While trajectory-ranking approaches are effective in single-agent reinforcement learning, they do not directly extend to multi-agent settings due to three fundamental challenges. First, in MARL, trajectory quality depends on the joint policy of all agents, making it impossible to evaluate or rank trajectories independently of other agents' behaviors. Second, the learning process is inherently non-stationary, as agents simultaneously update their policies, causing the distribution of trajectories (and their induced rankings) to shift over time. Third, unlike the single-agent case, MARL lacks a global notion of optimality: there is no universally optimal policy or trajectory ordering, but rather a set of interdependent best-response solutions. As a result, the theoretical guarantees underlying SORS, which rely on consistent global trajectory rankings, no longer hold in this setting. These challenges necessitate a fundamentally different formulation of policy invariance for trajectory-based reward shaping in MARL.

Our contributions are threefold:

1. **A multi-agent formulation of policy invariance for trajectory-based reward shaping.** The standard trajectory-ranking guarantees from the single-agent setting do not directly apply in MARL due to joint policy dependence and the absence of a global optimality notion. To address this, we reformulate policy invariance through conditional best-response reasoning, proving that total-order-equivalent shaping rewards preserve each agent's best-response set under fixed opponent policies, and consequently preserve the set of Nash equilibria. Section 4 presents this result.

2. **ARMS: a self-supervised reward shaping framework for MARL.** We propose ARMS, an online reward learning method for MARL that leverages trajectory ranking while accounting for the non-stationary and coupled nature of multi-agent learning. The framework learns a shared shaping

[1]The return is the cumulative reward obtained in a reinforcement learning problem.

signal across agents, and its design is motivated by the conditional best-response guarantee above. Section 5 introduces the framework.

3. **Empirical analysis of reward learning under multi-agent dynamics.** We show that ARMS improves sampling efficiency as the number of agents increases and as rewards become sparser, while also generalizing to unseen environments. In addition, we identify a MARL-specific failure mode in which coupled policy–reward dynamics can induce oscillatory behavior under limited exploration, causing the shaping model to reinforce suboptimal cyclic behaviors. We show that increasing exploration mitigates this effect and stabilizes learning. Section 6 details these experiments.

In our experiments, we use a grid-world multi-agent pathfinding domain to enable interpretable analysis, since coordination behaviors are directly observable through agent interactions and solution quality is readily quantifiable. Each agent only observes local information, so the environment is modeled as a partially observable Markov game. Finally, we evaluate ARMS with two MARL backbones, IPPO and MAPPO, demonstrating that its benefits are not specific to a single base algorithm.

1.1 Paper Organization

The remainder of the paper is organized as follows: Section 2 reviews related works, Section 3 provides the necessary terminology and notation used in this work, specifically for multi-agent reinforcement learning (MARL) and Reward Shaping, Section 4 formalizes reward function equivalence and Nash Equilibria invariance in the multi-agent setting and establishes its role within the proposed framework, Section 5 introduces the ARMS framework in detail and provides its pseudo-code implementation, Section 6 presents empirical results demonstrating the effectiveness of ARMS against no-shaping and potential-based reward shaping (PBRS) in multiple sparse-reward settings across different RL backbones, and provides evidence that increased exploration mitigates premature convergence to suboptimal behaviors. Section 7 concludes the paper and discusses future work.

2 Related Works

2.1 Reward Shaping in SARL and MARL

In single agent reinforcement learning, Reward shaping refers to the modification of the reward signal to accelerate learning. The modification is typically an incorporation of prior-knowledge into the task which reduces the number of suboptimal actions made to select the optimal action (Randløv & Alstrøm, 1998; Ng et al., 1999).

One way of doing so is by introducing an additive term to the original reward function. Let $\mathcal{R}(s, a, s') \in \mathbb{R}$ denote the reward function of the MDP underlying the RL problem, where the input tuple (s, a, s') indicates the state-action-next state and let $F(s, a, s') \in \mathbb{R}$ denote a shaping term. Then it is replaced with

$$\mathcal{R}'(s, a, s') \stackrel{\text{def}}{=} \mathcal{R}(s, a, s') + F(s, a, s')$$

Reward shaping is risky because it can change the set of optimal policies. To deal with this problem potential-based reward shaping (PBRS) was proposed (Ng et al., 1999) which keeps the set of optimal policies unchanged. The form of the shaping term is simply:

$$F(s, a, s') = \gamma\phi(s') - \phi(s)$$

where γ is the discount factor in the RL problem and $\phi(s) \in \mathbb{R}$ is any function.

Memarian et al. (2021) proposes Self-Supervised Online Reward Shaping (SORS), which completely replaces the reward function with the shaped term:

$$\mathcal{R}'(s, a, s') = F_\theta(s, a, s')$$

where F_θ is the shaping term parameterized by θ. The framework alternates between two phases: (a) a reinforcement learning phase, in which the agent is trained using the updated reward function, and (b) a reward update phase, in which the reward function is refined based on collected experiences during phase (a). The justification for completely replacing the original reward is that, if two reward functions define the same ordering over pairs of trajectories, then the two reward functions admit the same set of optimal policies (see Theorem 4.1 for a formal statement).

In the multi-agent setting, each agent has its own policy[2] and thus instead of learning an optimal policy, a compromise must be made and so agents typically learn a Nash equilibrium (Nash, 1951). The result on potential-based reward shaping (PBRS) was extended to these settings and shown to not alter the Nash-Equilibrium (Devlin & Kudenko, 2011).

2.2 Learning a Reward Function from Preference Ranking

Several prior works consider learning an inferred reward function from human preferences or ranking over trajectories, in the single-agent setting.

One work, Christiano et al. (2017), fits a reward function based on human preferences. More concretely, they use a policy to generate pairs of trajectories and query the human on their preferences. The RL algorithm uses the updated reward network to train its agent. And Ibarz et al. (2018) build on the aforementioned work to use an initial set of expert trajectories to train an initial policy rather than start from a random policy.

Another work, Self Supervised Online Reward Shaping (SORS) mentioned earlier, which uses an adaptation of the T-REX algorithm (Brown et al., 2019) for its reward inference part of the algorithm, learns a parameterized reward function online by ranking pairs of trajectories based on the original (sparse) reward. The reward network is then trained on the binary classification loss over these preferences. They show that their method collects higher reward than a baseline method throughout training (specifically Soft-Actor Critic) on a set of reward-sparse tasks.

Finally, Bui et al. (2025) also addresses sparse-reward MARL by converting episodic outcomes into trajectory preferences and using them to learn an implicit reward model within a value-decomposition and dual-advantage MAPPO framework. In contrast, our method learns shaping rewards directly from trajectory ordering, and its central theoretical contribution is different: as shown in Section 4, we establish a game-theoretic invariance result showing that, under total-order equivalence, best responses and therefore the set of Nash equilibria are preserved. This invariance result is the main theoretical motivation for our algorithmic design.

3 Background and Definitions

3.1 Partially Observable Multi-Agent Reinforcement Learning

We consider the Partially Observable Markov Decision Process (POMDP) in which multiple agents must make decisions simultaneously to maximize their ego-centric rewards:
We denote the multi-agent reinforcement learning (multi-agent RL) with the tuple $M \stackrel{\text{def}}{=} < \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{O}, \gamma >$, in which:

- $\mathcal{I} = \{1, \ldots, N\}$ is the set of agents,
- $\mathcal{S}$ is the finite set of states of the environment.
- $\mathcal{A} = \{\mathcal{A}_1 \ldots, \mathcal{A}_N\}$ is the finite set of actions for all agents,

[2]Generally, agents have distinct policies; parameter sharing uses a single shared parameterization across agents.

- A set of reward functions $\mathcal{R} \stackrel{\text{def}}{=} \{\mathcal{R}_1, \ldots, \mathcal{R}_N\}$. Concretely, $\mathcal{R}_i : \mathcal{S} \times A \times \mathcal{S} \to \mathbb{R}$ is the reward function for agent i where $A \stackrel{\text{def}}{=} \mathcal{A}_1 \times \ldots \times \mathcal{A}_N$ where $\mathbf{a} \in A$ denotes the **joint** actions across N agents,
- $\mathcal{T} : \mathcal{S} \times A \times \mathcal{S} \to [0, 1]$ is the transition probability which defines the environment dynamics, for which $\mathcal{T}(s, \mathbf{a}, s')$ to be interpreted as the probability to transition from state s to state s' when the joint action $\mathbf{a}$ was taken,
- A set of observation functions $\mathcal{O} \stackrel{\text{def}}{=} \{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$. Concretely agent i's observation function is given by $\mathcal{O}_i : A \times \mathcal{S} \times \mathcal{O}_i \to [0, 1]$ where $\mathcal{O}_i(s, \mathbf{a}, o)$ denotes the probability that agent i observes o_i given that the environment was in state s and the joint action $\mathbf{a}$ was taken,
- and $\gamma \in [0, 1]$ is the discount factor, which determines the relative importance of future rewards per agent [3]. We assume a common discount factor shared by all agents.

At each timestep $t \in \{1, \ldots, T\}$, where T denotes the terminal timestep, the environment generates a state of the environment $s_t \in \mathcal{S}$, and each agent $i \in \mathcal{I}$ receives its observation $o_t^i \in \mathcal{O}_i$ with probability given by its observation function $\mathcal{O}_i(\mathbf{a}_{t-1}, s_t, o_t^i)$ and selects an action $a_t^i \in \mathcal{A}_i$ (agent i's selection at timestep t). All agent selections at timestep t form the joint action $\mathbf{a}_t = (a_t^1, \ldots, a_t^N) \in A$. This joint action leads to a (stochastic) change in the environment generating a new state s_{t+1} with probability $\mathcal{T}(s_t, \mathbf{a}_t, s_{t+1})$. The environment also produces a *joint reward* $\mathbf{r}_t = (r_t^1, \ldots, r_t^N) \in \mathbb{R}^N$, where r_t^i denotes the reward received by agent i at timestep t.

Moreover, at each timestep t, each agent selects its action a_t^i with probability given by its stochastic policy $\pi_i(a_t^i|h_t^i)$, where $h_t^i \stackrel{\text{def}}{=} (o_0^i, \ldots, o_t^i)$, called the agent's *history*, is all of the agent's observation up to timestep t.

The return for agent i in a trajectory $\tau = (s_1, \mathbf{a}_1, \mathbf{r}_1, s_2, \mathbf{a}_2, \mathbf{r}_2, \ldots, s_T)$ is defined as

$$\mathcal{R}_i(\tau) \stackrel{\text{def}}{=} \sum_{t=1}^{T} \gamma^{t-1} \mathbf{r}_t^i.$$

Depending on the context, the state variables s_t in τ may be replaced by the corresponding agent observations.

A **reward-free (joint) trajectory** is defined as $\tau \stackrel{\text{def}}{=} (s_1, \mathbf{a}_1, s_2, \mathbf{a}_2, \ldots, s_T)$. When it is clear the trajectory doesn't include rewards, we just refer to τ as a (joint) trajectory, and define the joint observations of agents at timestep t to be $\mathbf{o}_t \stackrel{\text{def}}{=} (o_t^1, \ldots, o_t^N)$.

To complete the definition of the POMDP, for any agent i define $a_0^i = \emptyset$ and $o_0^i \stackrel{\text{def}}{=} \emptyset$.

With a slight abuse of notation we sometimes use $\mathcal{T}(s'|s, \mathbf{a})$ to refer to $\mathcal{T}(s, \mathbf{a}, s')$. This emphasizes that the selection of the next state is dependent on the previous action and previous state. In a similar fashion, we sometimes use $\mathcal{O}_i(o_t^i|\mathbf{a}_{t-1}, s_t)$ instead of $\mathcal{O}_i(\mathbf{a}_{t-1}, s_t, o_t^i)$.

3.2 Reward Shaping

Given a MARL instance $< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{O}, \gamma >$, *reward shaping* is the process of altering the original reward function for each agent to obtain a new (shaped) reward function either by replacing the original reward or augmenting it.

More concretely, let $\mathcal{R}_i(s, \mathbf{a}, s')$ be agent i's reward function and let $F_{i,\theta}(s, \mathbf{a}, s')$ be a shaping term for agent i parametrized by a shared θ across agents. Reward shaping may then be defined either as an additive modification,

$$\mathcal{R}_{i,\text{shaped}}(s, \mathbf{a}, s') = \mathcal{R}_i(s, \mathbf{a}, s') + F_{i,\theta}(s, \mathbf{a}, s'),$$

[3] Specifically, rewards obtained t time steps into the future are weighted by a factor of γ^t, thereby encouraging either short-term (γ close to 0) or long-term (γ close to 1) planning.

or as a replacement of the original reward,

$$\mathcal{R}_{i,\text{shaped}}(s, \mathbf{a}, s') = F_{i,\theta}(s, \mathbf{a}, s').$$

While training multiple agents in parallel it may encode the relationship between agent interactions implicitly, specifying how each agent should interact based solely on its local observations irrespective of its identity. Although $F_{i,\theta}$ is modeled as a function of the state-action-next state triple, in practice it will depend only on the local agent observations and actions. Finally, while the main goal of reward shaping is speeding up learning, using it may change the learned policy.

4 Nash Equilibria Invariance

In a previous work, it is shown that in single-agent reinforcement learning it is possible to define a preference oracle over trajectories, which is a binary relation $\leq_p$ over trajectories. With a preference oracle p, one can rank trajectories:

$\tau_1 \leq_p \ldots \leq_p \tau_k \leq_p \ldots$, where the preference oracle can be defined by any deterministic reward function $\mathcal{R}$:

$\tau_i \leq_{\mathcal{R}} \tau_j \iff \mathcal{R}(\tau_i) \leq \mathcal{R}(\tau_j)$.

Define two deterministic reward functions $\mathcal{R}, \mathcal{R}'$ "total-order equivalent" if and only if for all trajectory pairs τ_i, τ_j:

$\tau_i \leq_{\mathcal{R}} \tau_j \iff \tau_i \leq_{\mathcal{R}'} \tau_j$.

Importantly, it is possible to show that if two deterministic reward functions $\mathcal{R}, \mathcal{R}'$ are total-order equivalent then they will induce the same set of optimal-policies. Let us formally restate the theorem:

Theorem 4.1 (Single-Agent Total Order Equivalency (Restated from SORS Memarian et al. (2021))). *Given a deterministic[4] reward-free (single agent) MDP $M =< \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma >$ if two reward functions $\mathcal{R}, \mathcal{R}'$ are total order equivalent, they will induce the same set of optimal policies.*

Next, we adapt this theorem to the MARL setting. The main idea of the proof is that if all agents policies are fixed except for agent i, then switching out that single agent's reward function $\mathcal{R}_i$ to a total-order equivalent $\mathcal{R}'_i$ induces the same set of optimal policies for that agent.

Theorem 4.2 (Multi-Agent Total Order Equivalency). *Consider the deterministic reward-free multi-agent POMDP $M =< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \gamma >$. Fix an agent i and fix the policies of all other agents π_{-i}.*

Let $M_i^{\pi_{-i}}$ denote the induced single-agent environment faced by agent i when other agents follow π_{-i}. Let $\mathcal{R}, \mathcal{R}'$ be total-order-equivalent over trajectories from $M_i^{\pi_{-i}}$ then they induce the same set of optimal policies for agent i in $M_i^{\pi_{-i}}$. Equivalently, the set of optimal policies for agent i against π_{-i} is invariant under replacing $\mathcal{R}$ with a total-order-equivalent $\mathcal{R}'$ deterministic reward functions.

Proof. The proof is structured by first showing that $M_i^{\pi_{-i}}$ is a deterministic single-agent MDP, and then apply Theorem 4.1.

When other agent policies are fixed, agent i is the only decision maker. Set $M_i^{\pi_{-i}} \stackrel{\text{def}}{=} < \tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathcal{T}}, \gamma >$, where $\tilde{\mathcal{A}} \stackrel{\text{def}}{=} \mathcal{A}_i$ since agent i is the only actor, the state representations at timestep t denoted $\tilde{s}_t$ is simply the augmented state alongside the history of all agents $j \neq i$ up to timestep t: $\tilde{s}_t \stackrel{\text{def}}{=} (s_t, h_t^{-i})$ - the histories are included to restore the Markov Property to the MDP since policies of agents $j \neq i$ may rely on histories to make an action selection, and finally $\tilde{\mathcal{T}}$ simply transforms (s_t, h_t^{-i}) to (s_{t+1}, h_{t+1}^{-i}) by computing all other agent actions under their fixed policy and advancing the state and histories deterministically. Observe that under these assumptions $\tilde{\mathcal{T}}$ is deterministic.

Since $M_i^{\pi_{-i}}$ is a deterministic reward-free (single-agent) MDP and $\mathcal{R}, \mathcal{R}'$ are total-order-equivalent over trajectories from $M_i^{\pi_{-i}}$, then Theorem 4.1 implies that $\mathcal{R}, \mathcal{R}'$ induce the same set of optimal policies in $M_i^{\pi_{-i}}$. □

[4]That is, the transition dynamics for the MDP are deterministic.

As an immediate consequence, preservation of each agent's best-response set implies preservation of the Nash equilibrium set.

Corollary 4.3 (Nash Equilibrium Invariance)**.** *Suppose that for each agent $i \in \mathcal{I}$ and for a fixed joint policy of other agents π_{-i}, the reward functions $\mathcal{R}_i$ and $\mathcal{R}'_i$ are total-order equivalent over trajectories in the induced single-agent environment $M_i^{\pi_{-i}}$. Then the induced games under $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ and $\{\mathcal{R}'_i\}_{i\in\mathcal{I}}$ have the same set of Nash equilibria.*

Proof. By Theorem 4.2, for every agent i and every fixed π_{-i}, the replacement $\mathcal{R}_i \mapsto \mathcal{R}'_i$ preserves the set of best responses. Hence each agent's best-response correspondence is unchanged. Since Nash equilibria are exactly the fixed points of these best-response correspondences, the set of Nash equilibria is unchanged. □

4.1 Design Decisions and Applying Reward Function Equivalence in this work

The single-agent notion of reward function equivalence does not transfer directly to MARL, since in the multi-agent setting the solution to the problem is no longer an optimal policy, but an equilibrium induced by mutual best responses. However, when the policies of the other agents are held fixed, each agent faces an induced single-agent problem. Under this conditional view, total-order-equivalent reward functions preserve each agent's best-response set, and, by Corollary 4.3, preserve the set of Nash equilibria when the equivalence condition holds for all agents. This is the theoretical principle underlying our shaping design. Next, we describe the concrete design choices that operationalize this guarantee.

In this work, we use a parameterized reward function F_θ, modeled as $F_\theta(s, \mathbf{a}, s') \equiv F_\theta(o^i, a^i)$, where o^i and a^i are agent i's observation and action. Each agent then defines its shaped reward by

$$\mathcal{R}_{i,\text{shaped}}(s, \mathbf{a}, s') \stackrel{\text{def}}{=} F_\theta(o^i, a^i).$$

While the theorem and corollary provide an equilibrium-preservation guarantee under exact total-order equivalence, in ARMS, we operationalize this through the reward-learning loss (See Equation 1, Section 5). Specifically, the loss encourages the learned shaping reward to be consistent with the preference ordering induced by the original reward. Since the loss is computed per agent, the reward function is optimized from an agent-wise perspective.

In our implementation all agents share the same policy network and the same reward-shaping parameters. By aggregating the resulting learning signal across agents, the shaping network is trained to infer rewards through agent-wise trajectory ordering. This shared and symmetric structure allows for more samples collected from various agents throughout training and makes scaling to larger agent counts easier, since a single network suffices regardless of the number of agents.

The choices above are specific to our implementation, but in principle, the reward shaper could process either each agent's observations separately or the joint observations of all agents. The former, which is our chosen setup, mirrors decentralized information flow, encourages reward primitives that generalize across agents, and, as noted above, increases the amount of training signal available to the reward-shaping objective, since each agent contributes its personal trajectory independently. The latter corresponds to a more centralized shaping design and is more suitable for learning a personalized reward per-agent or as a common reward for all agents[5] which we leave as future work. Since the learned reward is used only during training, both choices are equally realistic to use; they differ primarily in the information structure used during reward-shaping optimization.

Finally, although the discussion above focuses on the design choices used in our implementation, Section 5 presents the most general formulation of ARMS without assuming a specific information flow in the reward-shaper functions.

[5]Common reward in MARL refers to the case where all agents receive the same reward scalar at each timestep.

Algorithm 1 ARMS: Automatic Reward-shaping in Multi-agent Systems

Require: MARL environment $M = \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \{\mathcal{R}_1, \ldots, \mathcal{R}_N\}, \mathcal{T}, \mathcal{O}, \gamma \rangle$.
Require: Length of each sampled trajectory L.
Require: Number of trajectory pairs to sample from buffer K.

```
Notation: |I| = N.
Initialize policy parameters {π_i}_{i=1}^N.
Initialize reward parameters {θ_i}_{i=1}^N.
Initialize trajectory-pair buffer D ← ∅.
while training not converged do
    // Reinforcement Learning Phase:
    Agents with policies {π_i}_{i=1}^N interact with the environment using shaped rewards {R_{θ_i}}_{i=1}^N, using any
    base RL algorithm for a fixed number of time-steps or episodes.
    Collect joint trajectories and store trajectory segments τ in buffer D.
    // Reward-Shaping Phase:
    Sample K pairs of trajectory segments (τ_k, τ_j) from D, each of length L.
    for each agent i ∈ I do
        Update θ_i by minimizing loss L_i(θ_i; D)      (Eq. 1)
    end for
end while
return  Learned policies {π_i}_{i=1}^N and shaped rewards {R_{θ_i}}_{i=1}^N
```

5 Method

We address the problem of sparse or delayed rewards in the multi-agent reinforcement learning (MARL) setting. Our key idea is to infer a dense reward functions for each agent that preserves the preference structure induced by the original sparse or delayed rewards. To this end, we introduce *Automatic Reward-shaping in Multi-agent Systems* (ARMS), a general reward-shaping framework that is agnostic to the underlying reinforcement learning algorithm and can be seamlessly integrated into existing MARL pipelines.

Let $M = \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_N\}, \mathcal{T}, \mathcal{O}, \gamma \rangle$ denote a MARL problem with N agents. Each agent $i \in \mathcal{I}$ is associated with a parameterized shaped reward function $\mathcal{R}_{\theta_i}$, which is learned online during training.

ARMS proceeds in two phases: a reinforcement learning phase and a reward-shaping phase. During the reinforcement learning phase, agents interact with the environment using their current shaped rewards $\mathcal{R}_{\theta_1}, \ldots, \mathcal{R}_{\theta_N}$ and collect experience in the form of joint trajectories.

Specifically, let

$$\tau_{t:t+L-1} = (\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1}, \mathbf{a}_{t+1}, \ldots, \mathbf{o}_{t+L-1})$$

denote a joint trajectory segment of length L, where $\mathbf{o}_t = (o_t^1, \ldots, o_t^N)$ is the joint observation at timestep t. Collected trajectory segments are stored in a trajectory buffer $\mathcal{D}$.

During the reward-shaping phase, K pairs of trajectory segments (τ_k, τ_j) are sampled from $\mathcal{D}$. For each sampled pair, every agent i performs a gradient update on its reward parameters θ_i using a binary classification loss that enforces consistency with the preference ordering induced by the original reward $\mathcal{R}_i$.

Formally, the loss used to train agent i's shaped reward function $\mathcal{R}_{\theta_i}$ is defined as

$$\begin{aligned}\mathcal{L}_i(\theta_i; \mathcal{D}) = -\sum_{(\tau_k, \tau_j) \sim \mathcal{D}} \Big[& \mathbb{I}(\tau_k(i) \leq_{\mathcal{R}_i} \tau_j(i)) \log P(\tau_k(i) \prec \tau_j(i)) \\ & + \big(1 - \mathbb{I}(\tau_k(i) \leq_{\mathcal{R}_i} \tau_j(i))\big) \log P(\tau_k(i) \succ \tau_j(i)) \Big]\end{aligned} \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function, and $\tau_k(i), \tau_j(i)$ denote the portions of the joint trajectories corresponding to agent i's observations and actions. This loss function has been used in other work to train a reward function with given pair-wise preference over trajectories (Memarian et al., 2021; Christiano et al., 2017).

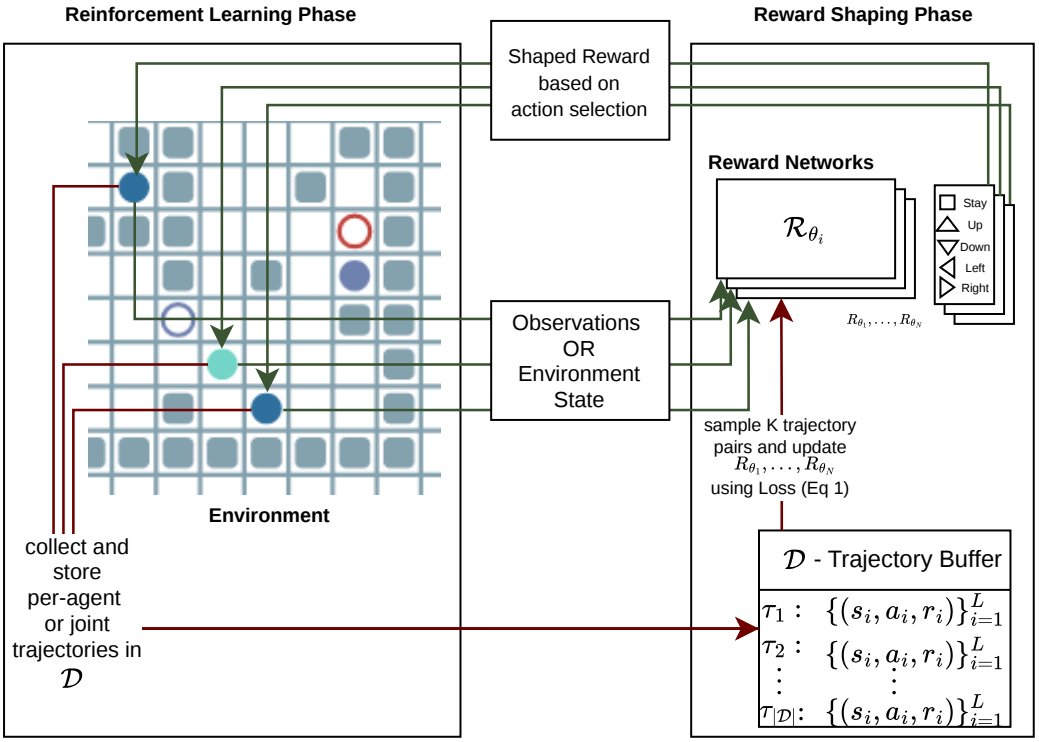


Figure 1: Overview of the ARMS framework. Green arrows correspond to operations in the reinforcement learning phase, while red arrows correspond to reward-learning operations.

The probability that trajectory τ_k is preferred over τ_j under the shaped reward is modeled using a softmax:

$$P(\tau_k \prec \tau_j) = \frac{\exp(\mathcal{R}_{\theta_i}(\tau_k))}{\exp(\mathcal{R}_{\theta_i}(\tau_k)) + \exp(\mathcal{R}_{\theta_i}(\tau_j))}. \tag{2}$$

Here, $\mathcal{R}_{\theta_i}(\tau)$ denotes the cumulative discounted shaped reward obtained by agent i over trajectory τ; for notational clarity we omit the agent index writing τ_k, τ_j to denote $\tau_k(i), \tau_j(i)$.

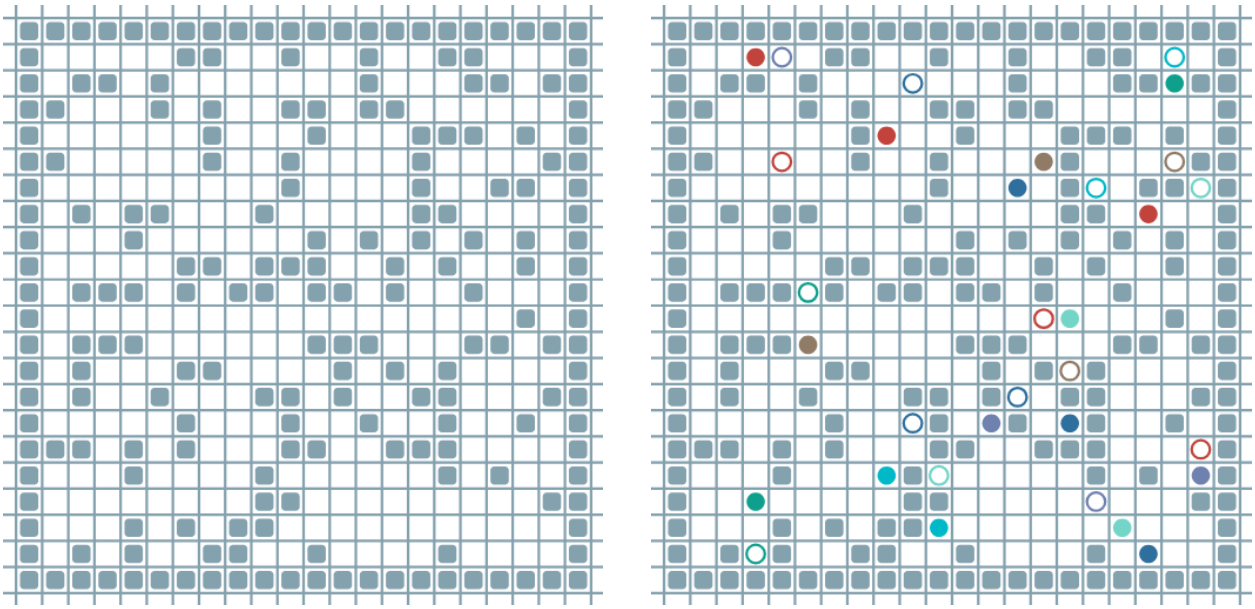
The full algorithm is provided in Algorithm 1 and Figure 1 shows the pipeline of ARMS schematically.

6 Experiments

6.1 Experiment Setup

We evaluate whether ARMS improves sampling efficiency and learning under sparse reward feedback, and whether it promotes coordinated behavior in cooperative multi-agent reinforcement learning. We test this in lifelong multi-agent path finding, where agents must repeatedly reach assigned goals while avoiding collisions. We compare ARMS with the base environment reward (in other words, with no reward shaping applied) and potential-based reward shaping across two MARL backbones, multiple agent counts, and reward-sparsity levels, and evaluate the generalization of the learned policies on 50 unseen maps.

The problem. We consider the following version of Lifelong Path-finding: several agents operate in a graph $G = (V, E)$ in which vertices correspond to locations and the edges to transitions between locations. Each agent starts in a given location and it needs to traverse the graph to reach its given goal location. When



(a) Empty map

(b) Map with 16 agents

Figure 2: The map used in our experiments with 30% obstacle density. Figure 2a displays the empty map, while 2b shows a typical episode initialization of 16 agents on the map; the filled circles denote agents while the hollow circles denote their corresponding targets.

an agent reaches its goal, it is then assigned a new goal location. In every timestep, all agents perform their actions simultaneously. Each agent can either stay in its current location or move to an adjacent one. Agents must cooperate as to not collide with or block one another. In our experiments the graph is a grid, and episodes are of fixed length $T_{\max}$. During training we set $T_{\max} := 256$.

The environment used in our experiments is the scalable and fast POGEMA (Skrynnik et al., 2025). One map is used in all our training experiments, a 20×20 map with 30% obstacle density is generated using POGEMA's random map generator, Figure 2 displays the map used. We evaluate the trained policy's sampling efficiency and quality on 50 maps not encountered during training - 40 random maps and 10 maze-like maps.

Agent observations are based on a recent Skrynnik et al. (2024)'s work. Each agent has a field of view (FOV) of its local environment. The view of radius $R \geq 0$ is the local $(2R+1) \times (2R+1)$ grid with the agent at the center. In our experiments we set $R := 5$, thus the FOV is of size 11×11. A variant of A^* is used to generate a global path for each agent to its target location, of which is encoded within its FOV as way-points. More specifically the agent's observations are split into two matrices: (1) One encoding shows the local obstacles within the FOV as -1 at that cell location, and the way-points along the computed path as $+1$, (2) The other, describing the local agents (including the self-agent placed at the center) as $+1$.

The environment reward. The reward is set to $r = +0.01$ each time the agent traverses into a way-point and 0 otherwise, making for a dense reward function. However, our goal is to evaluate over a sparse-reward environment. For this reason, we change the reward to instead accumulate over several time-steps and give the agent the accumulated reward all at once; we use the terms *original reward*, *base reward*, and *environment reward* interchangeably to refer to the reward provided by the environment, before adding any reward-shaping term. In the plots, this setting is denoted by *No*, indicating that no reward shaping is applied. In all of our experiments we set the delay at 20 time-steps, except in the second set of experiments in Section 6.2 in which we vary the delay period to show the robustness of our method across sparser rewards.

The reward network structure. The input to the network is the observations of a single agent. It is comprised of two parts: (1) The Spatial encoder is a ResNet (Xu et al., 2024) with an additional Multi-Layer

Perceptron (MLP) in the output layer of size 512. (2) Two additional MLP heads of size 128 followed by a SiLU activation each. (3) The output layer is an MLP head producing a 5-dimensional vector, passed through a tanh activation, where each component represents the reward associated with selecting the corresponding action. The training is end-to-end in all our experiments.

Base RL algorithms. We use two multi-agent reinforcement learning backbones: independent PPO (IPPO) and multi-agent PPO (MAPPO) (Schulman et al., 2017; de Witt et al., 2020; Yu et al., 2022). In both cases, the network follows the same architecture: a spatial encoder first processes the local observation, its output is passed to a GRU head modeling the observation history, and the resulting representation is fed into actor and critic heads. The actor outputs a distribution over the five available actions. During training, rollouts consisting of observations, actions, and rewards are gathered asynchronously from multiple parallel environments with a fixed number of agents. The difference between IPPO and MAPPO is in the critic used for optimization: IPPO trains agents using decentralized value estimates, whereas MAPPO uses a centralized critic during training while preserving decentralized execution.

Reward-shaping variants. Throughout the experiments, we compare three reward settings. The first, denoted "No" in the plots, trains the base MARL algorithm directly on the sparse original environment reward. The second, denoted "PBRS" (Potential Based Reward Shaping), augments the original reward with a hand-designed potential-based reward-shaping term, namely the commonly used manhattan distance in grid-world based environments. The third, denoted **ARMS**, which is our reward-shaping signal. In all cumulative-reward plots, we report the accumulated original dense environment reward, before applying reward sparsification.

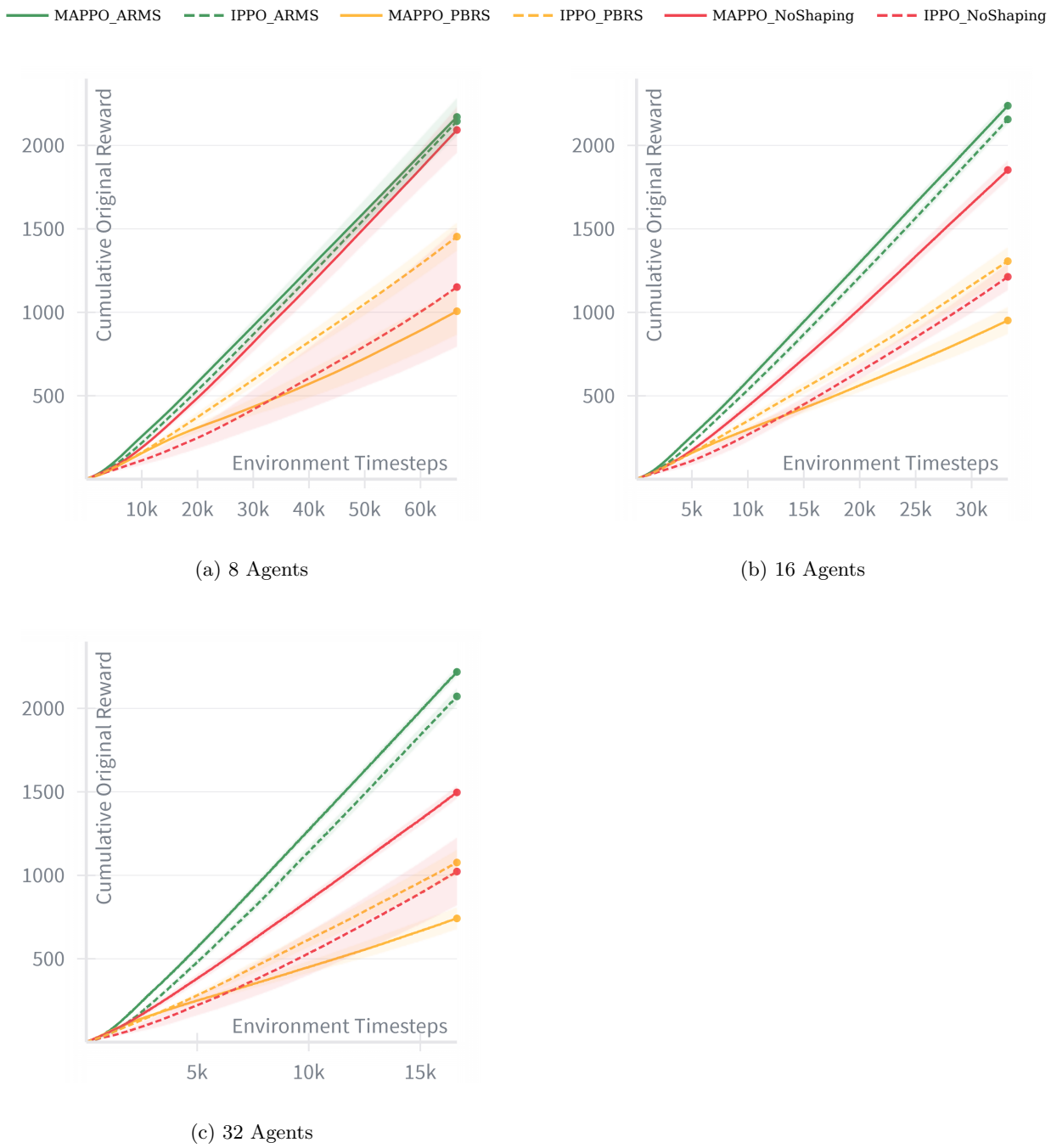
Training Plots. Training commenced across 8 parallel environments, each episode lasting $T_{\max} := 256$ timesteps. The cumulative reward plots in the following Sections show the average reward obtained across all agents throughout all 8 parallel environments at that time-step, and they span several episodes.

Evaluation. After training, the learned models are collected and evaluated for episodes of length $T_{\max} := 256$. We evaluate three aspects of the learned policies. (1) First, we measure policy quality on the training map using average throughput, defined as the number of times agents reach their goals divided by the episode length, aggregated across all agents. This metric captures the extent to which the learned policies solve the lifelong path-finding task. (2) Second, we evaluate whether the learned policies generalize beyond the training map by measuring cumulative original dense environment reward on the aforementioned 50 unseen maps. (3) Finally, we measure agent coordination on the same 50 unseen maps by reporting the normalized number of collisions accumulated during evaluation. Sections 6.3 and 6.4 detail these results.

Hyperparameters. All the base-algorithm hyper-parameters relating to Skrynnik et al. (2024) we keep unchanged, including the radius $R := 5$ mentioned earlier. The only exception is PPO's exploration coefficient discussed in more detail in Section 6.4. The size and number of MLP heads in the reward network were optimized while holding all other parameters fixed, however the initial guess also outperformed the base algorithm. We additionally scale the reward from $[-1, +1]$ to $[-0.1, +0.1]$, which we observe to have greatly increased the performance of the network in terms of total accumulated original reward throughout training. Little to no tuning was done to the hyper-parameters discussed next. First, we cap the trajectory buffer size at $16,384$ and set $K := 8,192$ which means 50% of the buffer is renewed at every Reward Shaping Phase (see Algorithm 1 step 9). Each trajectory was of size $L := 16$ time-steps making several trajectories uninformative for the neural-network in the cases the reward delay was set to $\in \{20, 30\}$. All experiments perform training for $\approx 4M$ agent-actions, thus when more agents are used, fewer timesteps (and therefore, fewer episodes) are present in the plots, for 16 agents that is about $\approx 1K$ episodes.

6.2 Sampling Efficiency

In this section, we evaluate whether ARMS improves sampling efficiency during training. We compare ARMS against no reward shaping and potential-based reward shaping (PBRS), using both IPPO and MAPPO as the underlying MARL backbone. Unless stated otherwise, the sparse environment reward is produced by accumulating the original dense reward over 20 timesteps and revealing the accumulated reward only at the end of the interval. All cumulative-reward curves report the original dense environment reward, before reward sparsification.



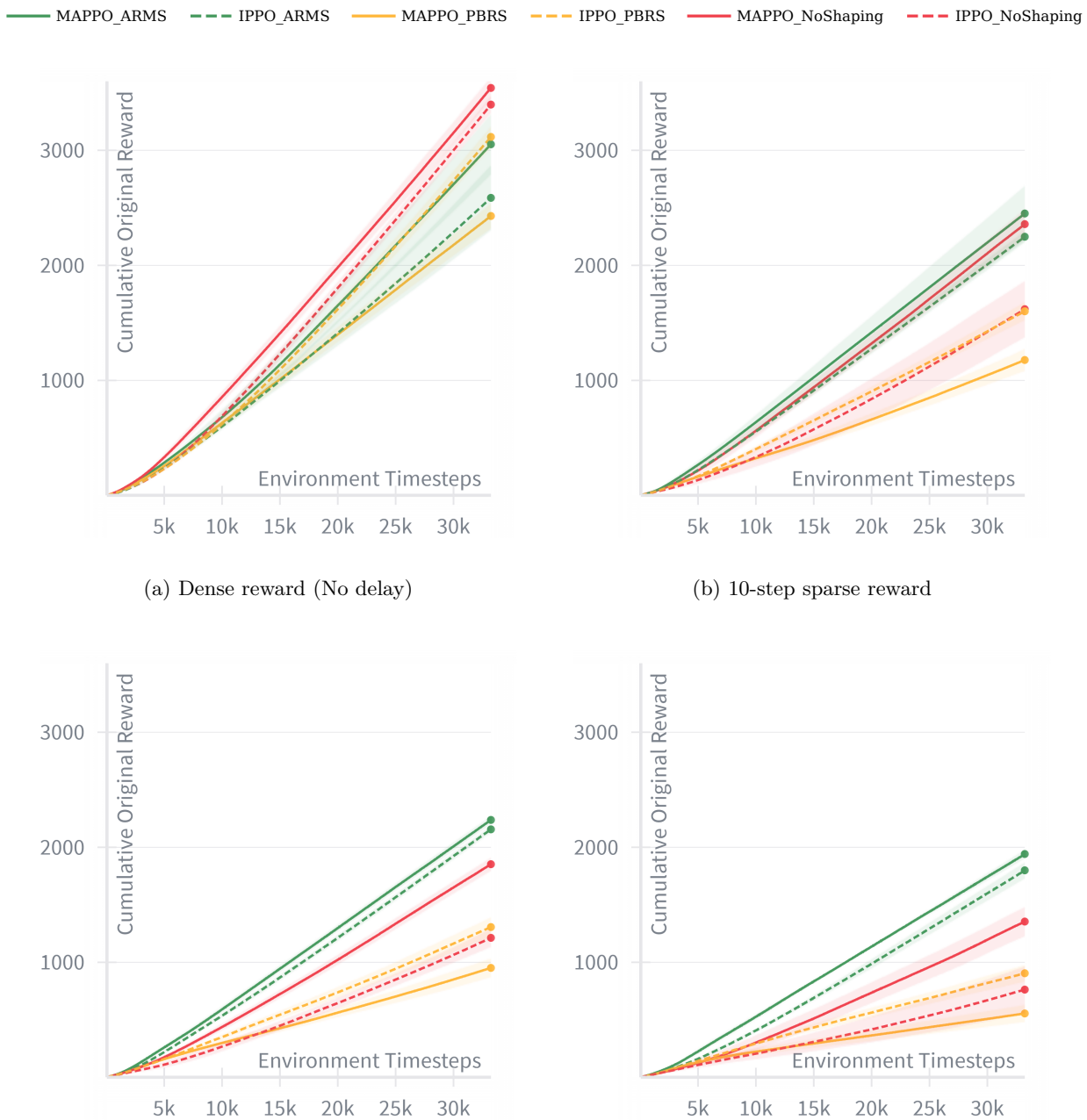
(a) 8 Agents

(b) 16 Agents

(c) 32 Agents

Figure 3: Learning curves under a 20-step sparse reward for 8, 16, and 32 agents. We compare ARMS, PBRS, and no reward shaping, each combined with IPPO and MAPPO. The reported curves are the mean across 10 seeds, no curve smoothing is applied, and the shaded region represents the standard deviation.

In the first set of experiments, we vary the number of agents while keeping the reward delay fixed at 20 timesteps. Figure 2b shows an example episode initialization with 16 agents, and Figure 3 reports the corresponding learning curves for 8, 16, and 32 agents. Across these settings, ARMS achieves higher cumulative original reward than PBRS and no shaping in the more challenging multi-agent regimes. For IPPO, the improvement is strong across all agent counts. For MAPPO, the improvement is especially pronounced for 16 and 32 agents, where coordination becomes increasingly important. This provides early



(a) Dense reward (No delay)

(b) 10-step sparse reward

(c) 20-step sparse reward

(d) 30-step sparse reward

Figure 4: Learning curves with 16 agents under different reward-sparsity levels. The dense setting corresponds to the original environment reward without delay. The sparse settings accumulate the original reward over $K \in \{10, 20, 30\}$ timesteps and reveal it only at the end of the interval. We compare ARMS, PBRS, and no reward shaping, each combined with IPPO and MAPPO. The reported curves are the mean across 10 seeds, no curve smoothing is applied, and the shaded region represents the standard deviation.

evidence that ARMS may be well-suited to induce inter-agent coordination in settings which have weak learning signals and harder multi-agent coordination demands, a performance category further explored in Section 6.4.

Next, we fix the number of agents at 16 and vary the reward-sparsity level. Figure 4 compares the dense original reward setting against sparse reward settings with accumulation intervals $K \in \{1 \text{ (dense)}, 10, 20, 30\}$. When the original reward is dense, all methods receive frequent task feedback, and ARMS does not provide an advantage, as expected. As the reward becomes sparser, ARMS provides a clearer benefit, achieving substantially higher cumulative original reward than PBRS and no shaping in the delayed-reward settings, while MAPPO is able to stay relatively competitive with ARMS in the 10 sparse delay setting but performance begins to degrade as rewards become sparser. This supports the main motivation of ARMS: learning a dense shaping signal is most useful when the environment reward itself becomes less frequent and less informative.

This result is notable because the reward-network trajectories have fixed length $L := 16$. Thus, when the reward delay is set to 20 or 30 timesteps, some sampled trajectories may contain no revealed environment reward. Nevertheless, ARMS is still able to learn useful shaping signals and improve training under these sparse-feedback regimes.

6.3 Avoiding Suboptimal Policy Convergence

In this set of experiments, we evaluate how to prevent ARMS into converging into suboptimal behaviour.

We show that varying the exploration coefficient has important consequences to the performance of the policy, which is especially difficult to learn in the MARL setting: agents continually co-adapt to each others behaviour and thus insufficient exploration can quickly lead to highly repetitive trajectories and stable but suboptimal coordination patterns. Particularily, in our chosen environment and underlying dense reward, agents can "hack" the reward by simply oscilitating repeatedly taking a positive small reward half the time and become unable to recover without enough exploration. This effect is especially pronounced in early training, agents will frequently collide or block one another during the RL phase (Algorithm 1, step 6), producing trajectories that explore only a narrow region of the joint observation space. As a result the reward is repeatedly trained on similar interaction patterns during the reward-shaping phase (Algorithm 1, step 9) and receives little signal about how alternative behaviors should be valued. Further, because agents share the policy parameters in our realization/implementation of ARMS, when agents block one another the policy is likely even more prone to this suboptimal behavior. As agents subsequently adapt their policies using this learned shaping reward (during the RL Phase), the system can enter a feedback loop where both the policy and the reward network specialize on this limited set of trajectories, making it difficult for the reward model to infer meaningful rewards for behaviors that have not yet ben observed. Increasing the exploration coefficient therefore encourages a more diverse trajectory distribution allowing the reward network to observe a wider range of interactions between agents and helping break this feedback loop. By measuring throughput we can measure whether the network has learned a meaningful policy with respect to solving the task.

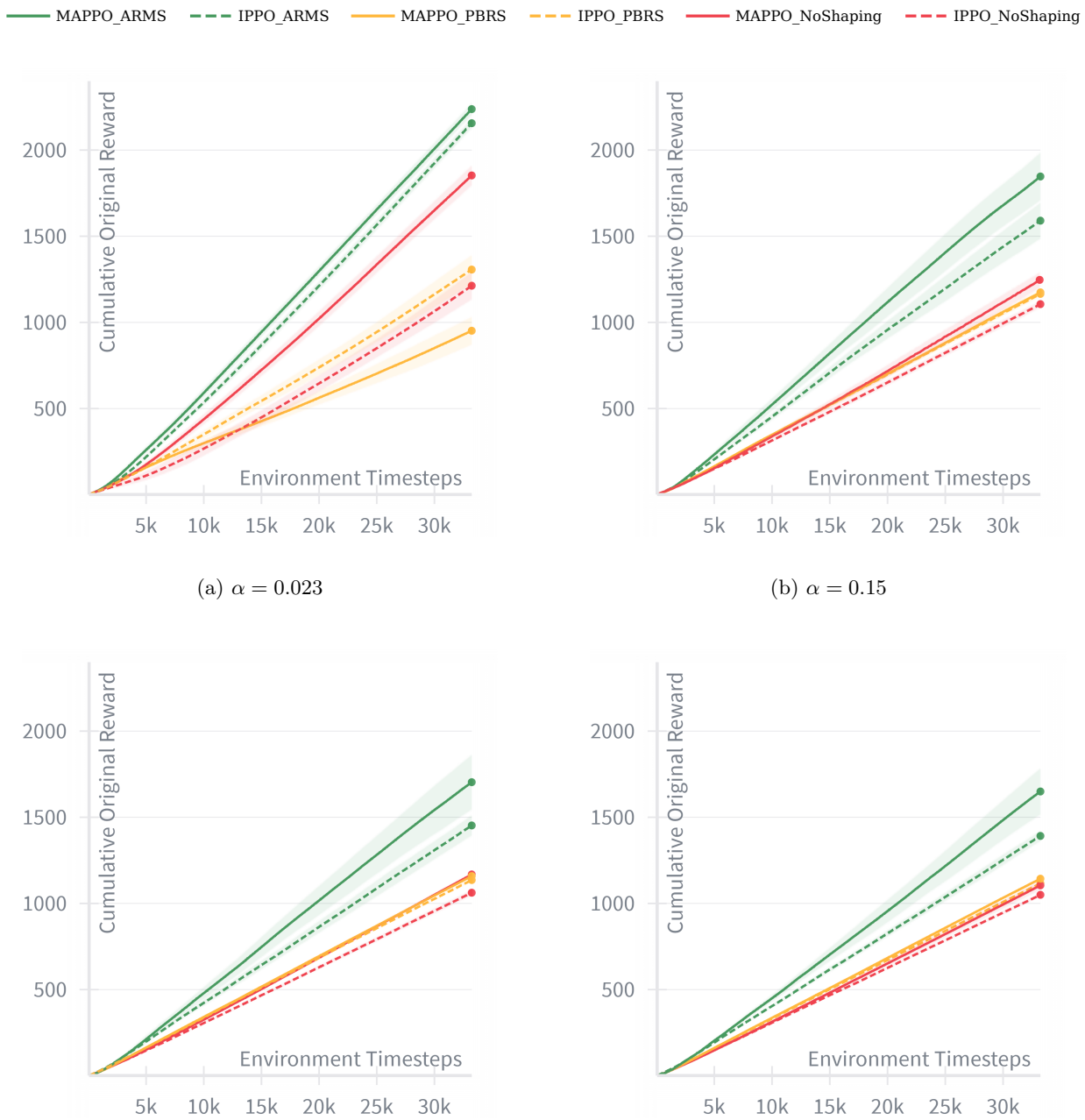
In all previous experiments, the exploration coefficient was kept fixed at 0.023. As shown in Figure 6, agents trained with the ARMS framework under this setting achieve nearly zero throughput across MAPPO and IPPO, indicating convergence to suboptimal policies.

First, we verify that ARMS continues to dominate in terms of reward accumulation, so we repeat the first set of experiments from Section 6.2 for 16 agents while varying the exploration coefficient. As Figure 5 reports, ARMS consistently outperforms the baselines in reward accumulation across all tested exploration values, showing robustness to this parameter, in agreement with the earlier results reported in Figure 3. As α increases, the obtained reward in the final timestep may be lower but as we can see in Figure 6, it mitigates this early convergence phenomenon.[6]

6.4 Agent coordination and generalization

In this set of experiments, we evaluate the generated policies, which were generated with the exploration coefficient set to 0.35, on 50 unseen maps during training - 40 random maps and 10 maze-like maps. First, we measure the accumulated original reward as we previously have across the episode, but also introduce another metric for the evaluation, *normalized collisions*, defined as the number of total collisions to occur divided

[6]The evaluated policies were the median performing policies in terms of cumulative reward at the end of training.



(a) $\alpha = 0.023$

(b) $\alpha = 0.15$

(c) $\alpha = 0.25$

(d) $\alpha = 0.35$

Figure 5: Effect of the exploration coefficient α on training performance. Across all tested values of α, ARMS maintains higher cumulative original reward than PBRS and no shaping.

by the episode length. This measure helps us further determine whether the reward teaches a coordinated policy. Recall that the original dense reward gives a positive reward for every step taken along the path, thus when agents are not colliding as much, they should be accumulating reward much more quickly, which is precisely what ARMS tries to teach the agents as we will see.

Figure 7 reports the cumulative original dense environment reward averaged over the 50 unseen maps, each episode ran for $T_{max} := 256$ timesteps. Across the evaluated agent counts, ARMS generally accumulates

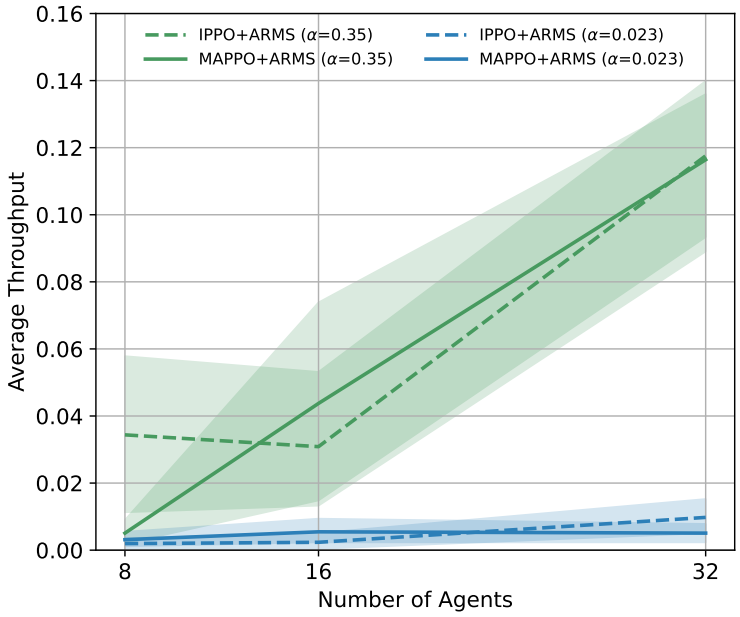


Figure 6: Average throughput of the learned ARMS policies under low and high exploration. We compare IPPO+ARMS and MAPPO+ARMS with $\alpha = 0.023$ and $\alpha = 0.35$ across different agent counts. The policies were evaluated across 100 seeds, and the shaded area represents the 95% confidence interval.

reward more quickly than PBRS and no shaping, indicating that the learned shaping signal improves policy quality beyond the training map.

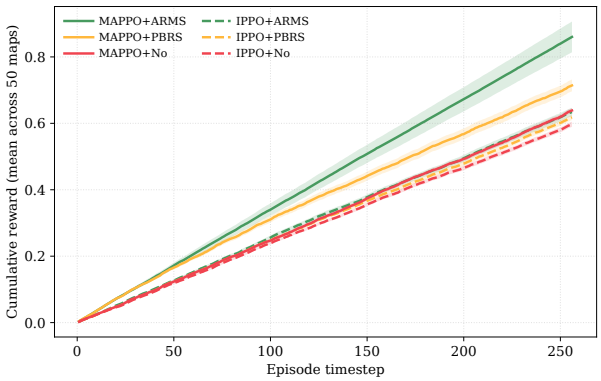
Figure 8 reports the normalized collisions accumulated over the same 50 unseen maps. Across all evaluated agent counts, ARMS produces substantially fewer collisions than PBRS and no shaping, suggesting that the learned shaping signal improves coordination rather than merely increasing reward accumulation.

We summarize the total accumulated dense reward throughout training for IPPO in Table 1, the throughput in Table 2, and the normalized collisions for IPPO in Table 3. Each entry reports mean ± standard deviation reported in the earlier Figures.

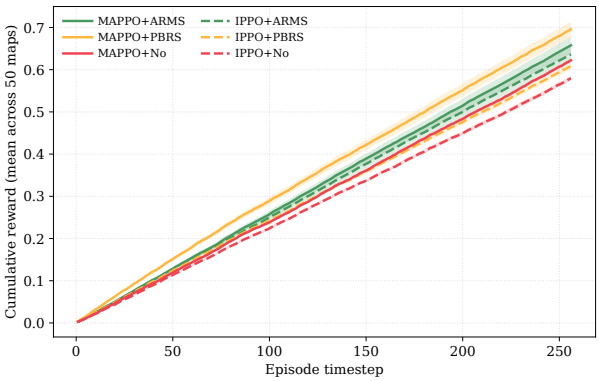
Table 1: Cumulative original dense (unshaped) reward on the training map for IPPO, broken down by (i) agent count at fixed reward sparsity = 20 and (ii) reward sparsity at fixed agent count = 16. Each cell reports mean±std observed in the final timestep. Columns compare the shaping methods No (unshaped), PBRS, and ARMS.

		Cumulative Original Reward		
		No	PBRS	ARMS
Agent Count	8	1114.58 ± 344.08	1413.82 ± 82.21	2116.54 ± 108.49
	16	1212.87 ± 79.48	1306.50 ± 83.41	2155.95 ± 28.06
	32	1022.83 ± 201.99	1076.61 ± 74.92	2071.69 ± 47.75
Reward Sparsity	10	1619.27 ± 244.68	1600.68 ± 78.23	2248.71 ± 47.97
	20	1212.87 ± 79.48	1306.50 ± 83.41	2155.95 ± 28.06
	30	762.55 ± 200.84	904.65 ± 71.63	1799.66 ± 70.29

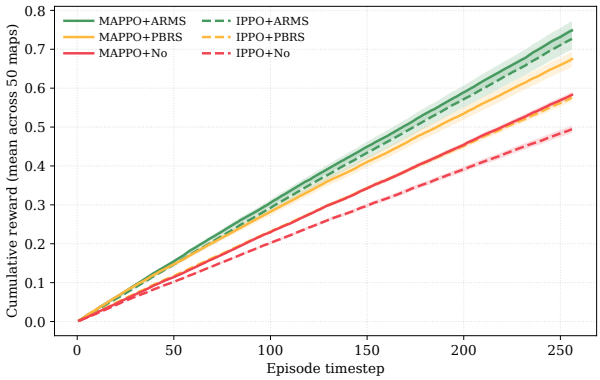
Finally, Figure 9 shows two example trajectories together with the rewards assigned along the path. Notably, the learned reward increases more sharply when the agent moves directly toward the goal, and encourages avoidance of other agents, and in this case receiving negative reward when it passes adjacently to them.



(a) 8 agents



(b) 16 agents



(c) 32 agents

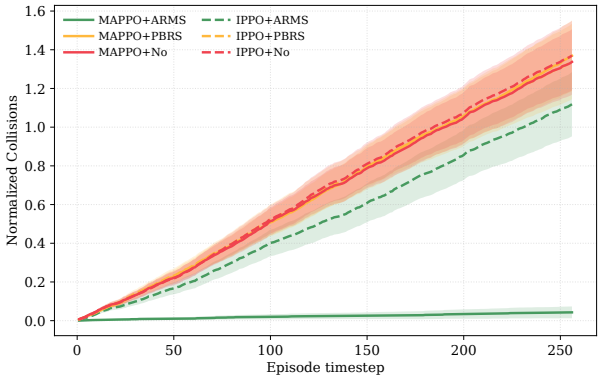
Figure 7: Cumulative original dense environment reward on 50 unseen evaluation maps for 8, 16, and 32 agents. The evaluated policies were trained with exploration coefficient $\alpha = 0.35$. We compare ARMS, PBRS, and no reward shaping with both IPPO and MAPPO.

Table 2: Average throughput on the training map (`random-Size20Map0`) for ARMS reward shaping, broken down by algorithm (IPPO, MAPPO), agent count (8, 16, 32), and exploration coefficient ($\alpha \in \{0.023, 0.35\}$). Each cell reports mean±std over 10 evaluation seeds.

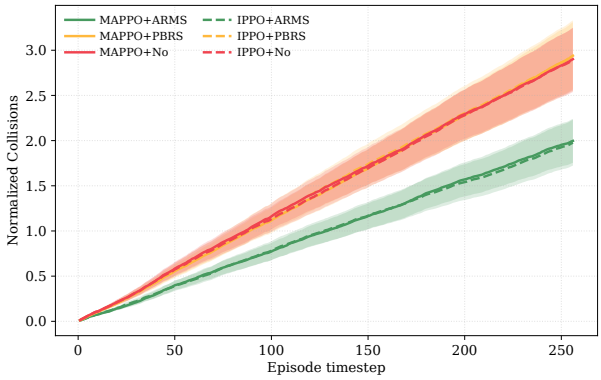
Algorithm	Count	$\alpha = 0.023$	$\alpha = 0.35$
IPPO	8	0.002 ± 0.003	0.034 ± 0.041
	16	0.002 ± 0.004	0.031 ± 0.036
	32	0.010 ± 0.009	0.118 ± 0.043
MAPPO	8	0.003 ± 0.004	0.005 ± 0.006
	16	0.006 ± 0.006	0.044 ± 0.053
	32	0.005 ± 0.005	0.116 ± 0.037

Table 3: Normalized collisions at the final timestep $T_{max} := 256$ on the 50-map out-of-distribution benchmark, for IPPO with $\alpha = 0.35$ and reward sparsity 20. Each cell reports mean±std across the 50 evaluation maps.

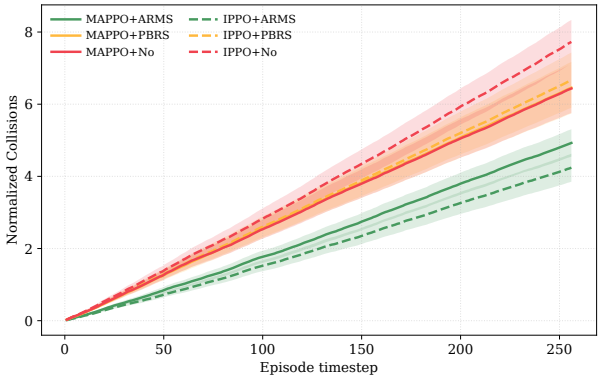
Agent Count	No	PBRS	ARMS
8	1.371 ± 0.690	1.360 ± 0.743	1.083 ± 0.594
16	2.875 ± 1.312	2.958 ± 1.424	1.955 ± 0.902
32	7.770 ± 2.231	6.619 ± 2.681	4.273 ± 1.448



(a) 8 agents



(b) 16 agents



(c) 32 agents

Figure 8: Normalized collisions accumulated on 50 unseen evaluation maps for 8, 16, and 32 agents. The evaluated policies were trained with exploration coefficient $\alpha = 0.35$. We compare ARMS, PBRS, and no reward shaping with both IPPO and MAPPO.

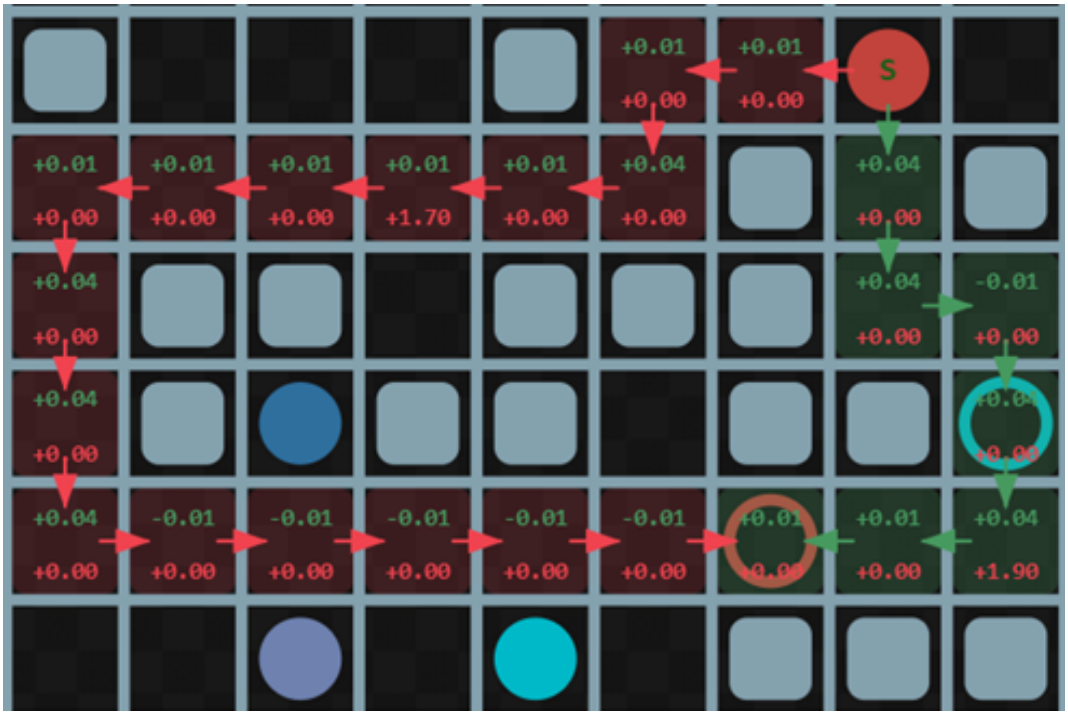


Figure 9: Two sample trajectories captured during an episode, along with the rewards the agent is expected to receive. All neighboring agents remain stationary in this example while the red agent traverses toward its goal. Green values indicate the learned reward (ARMS) while red values indicate the original sparse reward. Since the agent has already partially progressed along its path, the sparse reward accumulated over the previous 20 timesteps is received at timestep 5. The rewards are scaled $\times 10$ for visual convenience. In the shorter trajectory, the agent reaches the goal exactly at timestep 7, causing the accumulated sparse reward to be delivered two cells away from the goal. The sum of ARMS' reward at the shorter path is $+0.20$ while the longer path $+0.22$, indicating that higher reward can be accumulated when taking shorter paths.

7 Conclusion and Future Work

In this work, we presented ARMS, an **A**utomatic **R**eward-shaping framework for **M**ulti-agent **S**ystems which is able to integrate with any underlying multi-agent reinforcement learning algorithm. It aims to automatically infer rewards in sparse-setting environments by using the agent's sparse reward as a supervision signal to learn a parameterized *dense* reward signal $\mathcal{R}_{\theta_i}$ for each agent i.

The framework relies on the baseline algorithm to collect trajectories, and improves the parameterized reward signal by ranking trajectories based on the sparse reward. We theoretically prove that if certain assumptions hold, replacing **each** agent's reward function with another preserves the set of Nash-equilibria.

In our implementation the policy network parameters are shared across agents. Combined with the conditional best-response analysis, this symmetry motivates sharing the reward-shaping parameters and training the shaping network using observations aggregated from all agents, effectively treating each agent as an interchangeable instance of a representative learner interacting with copies of itself.

Empirically, ARMS improves sampling efficiency in sparse and delayed reward settings and yields policies with higher cumulative reward. The resulting policies also complete tasks more frequently compared to the baseline. We further observe that the learned policies generalize better to unseen maps, outperforming the baseline across a diverse set of evaluation environments.

Future work may explore allowing the reward network to condition on agent histories. In the current implementation, the shaping function produces a reward at each timestep based only on the agent's local observation and action. While this may be sufficient if some temporal information is implicitly captured by the network parameters, incorporating sequence models that encode observation histories could allow the shaping function to capture temporal dependencies that are not observable from a single timestep.

Future work could also investigate the stability of the reward learning process, including the interaction between trajectory ranking, reward delay, and policy optimization dynamics.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Carla E. Brodley (ed.), *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004. doi: 10.1145/1015330.1015430. URL https://doi.org/10.1145/1015330.1015430.

Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL https://www.marl-book.com.

Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 783–792. PMLR, 2019. URL http://proceedings.mlr.press/v97/brown19a.html.

The Viet Bui, Tien Mai, and Thanh Hong Nguyen. Preference-guided learning for sparse-reward multi-agent reinforcement learning. *ArXiv*, abs/2509.21828, 2025. URL https://api.semanticscholar.org/CorpusID:281658371.

Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11), 2021. ISSN 2076-3417. doi: 10.3390/app11114948. URL https://www.mdpi.com/2076-3417/11/11/4948.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural*

Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 4299–4307, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Christian Schröder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *CoRR*, abs/2011.09533, 2020. URL https://arxiv.org/abs/2011.09533.

Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In Liz Sonenberg, Peter Stone, Kagan Tumer, and Pinar Yolum (eds.), *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3*, pp. 225–232. IFAAMAS, 2011. URL http://portal.acm.org/citation.cfm?id=2030503&CFID=69153967&CFTOKEN=38069692.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8022–8034, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/8cbe9ce23f42628c98f80fa0fac8b19a-Abstract.html.

Xiaosong Lu, Howard M. Schwartz, and Sidney Nascimento Givigi. Policy invariance under reward transformations for general-sum stochastic games. *J. Artif. Intell. Res.*, 41:397–406, 2011. doi: 10.1613/JAIR.3384. URL https://doi.org/10.1613/jair.3384.

Farzan Memarian, Wonjoon Goo, Rudolf Lioutikov, Scott Niekum, and Ufuk Topcu. Self-supervised online reward shaping in sparse-reward environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pp. 2369–2375. IEEE, 2021. doi: 10.1109/IROS51168.2021.9636020. URL https://doi.org/10.1109/IROS51168.2021.9636020.

John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. ISSN 0003486X, 19398980. URL http://www.jstor.org/stable/1969529.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In Ivan Bratko and Saso Dzeroski (eds.), *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pp. 278–287. Morgan Kaufmann, 1999.

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications. *CoRR*, abs/1812.11794, 2018. URL http://arxiv.org/abs/1812.11794.

Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.*, 53(11):13677–13722, 2023. doi: 10.1007/S10489-022-04105-Y. URL https://doi.org/10.1007/s10489-022-04105-y.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In Jude W. Shavlik (ed.), *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pp. 463–471. Morgan Kaufmann, 1998.

Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 21:178:1–178:51, 2020. URL https://jmlr.org/papers/v21/20-081.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

Alexey Skrynnik, Anton Andreychuk, Maria Nesterova, Konstantin S. Yakovlev, and Aleksandr Panov. Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 17541–17549. AAAI Press, 2024. doi: 10.1609/AAAI.V38I16.29704. URL https://doi.org/10.1609/aaai.v38i16.29704.

Alexey Skrynnik, Anton Andreychuk, Anatolii Borzilov, Alexander Chernyavskiy, Konstantin S. Yakovlev, and Aleksandr Panov. POGEMA: A benchmark platform for cooperative multi-agent pathfinding. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=6VgwE2tCRm.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar (eds.), *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL http://dl.acm.org/citation.cfm?id=3238080.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction, 2nd Edition*. MIT Press, 2018. URL http://www.incompleteideas.net/book/the-book-2nd.html.

Annie Wong, Thomas Bäck, Anna V. Kononova, and Aske Plaat. Deep multiagent reinforcement learning: challenges and directions. *Artif. Intell. Rev.*, 56(6):5023–5056, 2023. doi: 10.1007/S10462-022-10299-X. URL https://doi.org/10.1007/s10462-022-10299-x.

Huazhi Xu, Xiaoyan Luo, and Wencong Xiao. Multi-residual unit fusion and wasserstein distance-based deep transfer learning for mill load recognition. *Signal Image Video Process.*, 18(4):3187–3196, 2024. doi: 10.1007/S11760-023-02981-6. URL https://doi.org/10.1007/s11760-023-02981-6.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Auton. Agents Multi Agent Syst.*, 38(1):4, 2024. doi: 10.1007/S10458-023-09633-6. URL https://doi.org/10.1007/s10458-023-09633-6.

A Appendix

You may include other additional sections here.