Adaptive Reward Sharing to Enhance Learning in the Context of Multiagent Teams

Kyle Tilbury, David Radke

Keywords: Multiagent Reinforcement Learning, Meta-Reinforcement Learning, Coordination

Summary

Real-world populations often include diverse social structures such as sub-groups or teams, creating heterogeneous incentives that complicate coordination. Therefore, autonomous agents must be able to adapt their individual incentives based on their surrounding population. To address this challenge, we introduce a decentralized multiagent reinforcement learning framework in which each individual agent learns to adapt both its behavior and its reward-sharing strategy within a defined social structure in mixed-motive environments. Inspired by meta-RL, each agent in our framework maintains two policies: a low-level behavioral policy and a high-level reward-sharing policy that updates its individual reward function, changing how agents distribute earned rewards and thereby shaping the incentives within the population. We demonstrate the viability of this self-tuning approach by showing how agent populations can learn to coordinate more effectively via the simultaneous adaptation of heterogeneous incentive configurations. This work is a step toward integrating learning agents into real-world scenarios with complex social structures and varying incentives.

Contribution(s)

1. We introduce a multi-level framework enabling individual agents to not only learn behavior, but also adapt heterogeneous reward-sharing parameters in complex environments.
   Context: Recent work often assumes homogeneous or fixed reward-sharing schemes, limiting agents' ability to adapt to evolving social contexts (Durugkar et al., 2020; Radke et al., 2023a). Other approaches let agents learn to share or gift rewards but they either lack social structure or assume a cooperative global objective (Lupu & Precup, 2020; Yi et al., 2022). Our work, which considers mixed-motive environments with social structure, allows agents to learn and adapt to heterogeneous reward-sharing schemes, better capturing more diverse dynamics with social structures and varying incentives.
2. We show that our framework enhances population-level coordination, overcoming sub-optimal initialization and surpassing non-adaptive fully-cooperative baselines.
   Context: Using a standard but sub-optimal fully cooperative initialization, we demonstrate that adapting reward-sharing parameters enables agents to exceed the performance of non-adaptive fully-cooperative baselines by 34.2% and 20.3% (in mean population reward) across our two evaluation environments. Additionally, inspecting agent behaviors in one environment reveals that our adaptive agents consistently learn the best observed joint policy identified in prior work (Radke et al., 2023a).
3. We demonstrate that the heterogeneous reward-sharing parameterizations learned by our framework are highly effective when used to train new agent populations.
   Context: Using the heterogeneous reward-sharing schemes discovered during online tuning (i.e., while agents dynamically update their reward-sharing parameters during learning) as static parameterizations for newly instantiated agents yields further performance gains in both evaluation environments over online tuning. In one environment, these new populations surpass the best known configuration from prior work, achieving the highest observed reward while maintaining significantly greater equality. Discovering effective heterogeneous configurations through exhaustive or heuristic search of existing methods is onerous, our approach leverages reinforcement learning to autonomously learn effective solutions.

Adaptive Reward Sharing to Enhance Learning in the Context of Multiagent Teams

Kyle Tilbury[1,†], **David Radke**[2,†]
ktilbury@uwaterloo.ca, dradke@blackhawks.com

[1]**University of Waterloo, Canada**
[2]**Chicago Blackhawks, USA**

[†] Equal contribution

Abstract

Real-world populations include diverse social structures, such as sub-groups or teams, that create heterogeneous incentives that complicate coordination. Agents learning to cooperate in these dynamic environments must be able to adapt their internal incentives with their surroundings. We address this challenge with a decentralized multiagent reinforcement learning framework in which each agent has two policies, a low-level behavioral policy and a high-level reward-sharing policy, to adapt both its behavior and its reward function within mixed-motive environments with social structure. We demonstrate the viability of our approach by showing that agents coordinate more effectively through this simultaneous adaptation of heterogeneous reward-sharing configurations. Empirically, our framework enhances population-level outcomes, overcoming sub-optimal initializations and surpassing non-adaptive fully-cooperative baselines in two evaluation domains. Furthermore, the heterogeneous reward-sharing parameterizations that our method learns prove highly effective when applied to new agent populations, yet identifying such configurations through exhaustive or heuristic search is burdensome in this complex search space. By enabling agents to learn and adapt to heterogeneous reward-sharing schemes, our work better captures more diverse dynamics with social structures and varying incentives.

1 Introduction

The study of cooperation is crucial in artificial intelligence, multiagent systems, and reinforcement learning (RL) (Dafoe et al., 2020; 2021). Individual learning agents that cooperate can enhance their capabilities beyond those of a single agent; however, agents often encounter mixed-motive scenarios with diverse incentives that complicate coordination (Leibo et al., 2017). Furthermore, real-world populations often contain diverse social structures, in the form of sub-groups or teams, that increase the complexity of incentive structures. Agents must be able to adapt and learn effective incentive configurations to operate successfully in these heterogeneous, dynamic settings.

Many existing approaches default to fully cooperative designs, where all agents share rewards equally, to facilitate learning in complex mixed-motive scenarios. However, recent work highlights the learning advantages of maintaining some mixed incentives (Durugkar et al., 2020; Radke et al., 2022; Roesch et al., 2024). Determining which mixture of individual and shared rewards is most effective depends on agents' roles, policy distributions, and broader environmental factors, resulting in a search space that quickly becomes large and unwieldy. This space can be searched by either defining some population structure (i.e., sub-groups) and modifying inter-agent social dependencies (i.e., degrees of reward sharing with groups) or defining reward-sharing methods and changing the

underlying structure of sub-groups. In this paper, we propose a multi-level learning algorithm inspired by meta-RL that autonomously navigates the search space of reward-sharing configurations for a given social structure, enabling agents to adapt the parameters of their internal reward functions in the context of other agents.

Our work is concerned with the *prescriptive* agenda of multiagent RL (MARL), focusing on the behaviors and performance of agents *during* learning (Albrecht et al., 2024). Specifically, we study the context of individual learning agents operating within a defined social structure (i.e., agents have membership to different groups or teams) with the ability to update their inter-agent social dependencies. Our agents consist of two internal RL policies that operate at different timescales. One policy operates within a multiagent environment, observing states and taking actions in the context of other agents (i.e., their behavioral policy). A second policy operates in the space of the behavioral policy's reward function, a meta-environment, by observing how the behavioral policy shares reward among different groups in the population and taking actions to update this reward-sharing scheme (i.e., their reward-sharing policy). This interaction is similar to problems in meta-RL, where the behavioral policy represents the "inner-loop" and the reward-sharing policy represents the "outer-loop" (Schmidhuber, 1987; Beck et al., 2023) to learn hyperparameters of the reward function that best support the overall learning objective. This paper makes the following contributions:

- We introduce a multi-level MARL framework wherein agents not only learn behavioral policies but also learn heterogeneous reward-sharing parameters within complex environments.
- We demonstrate that this framework enhances population-level coordination, overcoming sub-optimal initializations and surpassing fully-cooperative non-adaptive baselines.
- We show that the heterogeneous reward-sharing schemes discovered via our approach continue to yield strong results when applied as static parameters in new agent populations.

2 Preliminaries

We are interested in the space of mixed-motive stochastic game environments that are not purely cooperative or competitive. Consider a stochastic game, defined as $\mathcal{G} = \langle \mathcal{N}, S, \{A\}_{i \in \mathcal{N}}, \{R\}_{i \in \mathcal{N}}, P, \gamma \rangle$, where $\mathcal{N}$ is the set of N individual learning agents. S represents the state space and s_i represents an observation of agent i. We assume that all agents are able to observe the environment the same (i.e., no agent has privileged observations). A represents the action space and R represents the reward function, where $R_i : S \times A \times S \mapsto \mathbb{R}$ and each agent receives the same reward from the environment for executing the same state-action pair (i.e., no particular agent receives *more nor less* reward for specific state-action pairs). $P : S \times A \mapsto \Delta(S)$ is the transition function and γ is the discount factor where $0 \leq \gamma < 1$. Additionally, we assume the existence of **reward-causing state-action (RCSA)** pairs, areas of the state space that do not carry explicit reward themselves but cause reward to be obtained elsewhere in the environment (Arjona-Medina et al., 2019; Radke et al., 2023b).

We define a *team* to be a collection of agents that can have some degree of common interest as a way to introduce social structure to the population. Similar to past work, we assume that agents are assigned to teams *a priori*, though agents are not required to share rewards with teams they belong to. A team to which agent i belongs is denoted $T_n \subset \mathcal{N}$ and agents may belong to any number of teams. The set of all teams in the system is defined as $\mathcal{T}$ with the number of teams being denoted $|\mathcal{T}|$. The set of all teams that agent i belongs to is defined by the set $\mathcal{T}_i$. Given these definitions, multiple teams may coexist in the same environment and may not be in zero-sum competition. For example, employees at a company can belong to multiple teams: an engineering team, a leadership team, and various social groups within the company. These employees have varying degrees of common interest with their peers in each group; however, the individual teams are likely not in zero-sum competition.

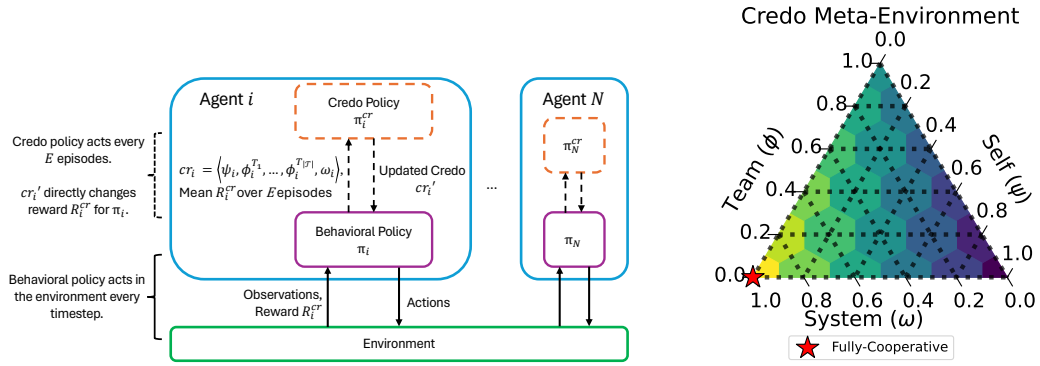
3 Related Work and Background

While sharing rewards has been shown to support cooperative behavior, reward sharing alone does not guarantee agents will learn the best joint policy. Recent work has formalized the relationship between the credit assignment problem in MARL and reward-sharing groups, motivating the need for specialized social structures or reward-sharing parameters among agents (Radke et al., 2023b). Other research has analyzed how the balance between individual and population-level rewards influences learning. The Social Value Orientation (SVO) model proposes a scale ranging from purely individualistic to fully altruistic reward functions (McKee et al., 2020) and Durugkar et al. (2020) investigate how specific blends of personal and group rewards can benefit coordination tasks; however, sharing schemes remain static throughout experiments in both models. D3C overcomes the static limitations of these frameworks by allowing agents to update the amount of reward they share with any other individual in the system through random perturbations (Gemp et al., 2022). However, the populations in these prior works lack social structure and agents are not able to be strategic in how they share rewards. In contrast, our work considers populations with defined social structures and allows agents to dynamically learn individual reward-sharing parameters for groups they belong to. Our agents use RL to update their reward-sharing policies; thus, agents are able to update diverse reward-sharing parameters strategically.

Other distinct reward-sharing mechanisms have been explored where agents explicitly act to share rewards. Reward *gifting* allows agents to directly transfer rewards to their peers (Lupu & Precup, 2020; Dong et al., 2021). Agents in the Learning to Incentivize Others (LIO) model learn gifting functions that modify the rewards given to other agents (Yang et al., 2020). However, the framework is not *budget balanced* and can introduce excess reward into the environment. Formal contracts have been shown to align agents on a reward-sharing scheme but only supports a single contract at once (Christoffersen et al., 2023). Alternative to these approaches, our work leverages meta-RL to dynamically update agent preferences for multiple groups through reward-sharing hyperparameters instead of targeted peer incentives.

Credo Background. Specifically, our work builds upon the *credo* model for sharing rewards between MARL agents in the context of social structure (Radke et al., 2023a). The credo model extends the reward-sharing framework to populations with social structures where agents can optimize their behavior for themselves, a subset of agents (i.e., a team), or the entire system population. The ratio of an agent i's reward shared with each group (self, teams, system) is regulated by credo *parameters*, represented as: $\mathbf{cr}_i = \langle \psi_i, \phi_i^{T_1}, \ldots, \phi_i^{T_{|\mathcal{T}|}}, \omega_i \rangle$. Credo parameters define a reward-sharing configuration of how much reward an agent shares among each group, where ψ is the credo parameter for i's individual reward IR_i, $\phi_i^{T_n}$ is the credo parameter for a team-based reward $TR_i^{T_n}$ from team T_n, and ω_i is the credo parameter for the system-based reward of the entire system SR. The credo parameters within $\mathbf{cr}_i$ always sum to one. Past work implements team rewards $TR_i^{T_n}$ and system rewards SR as the mean reward obtained by agents in T_n and $\mathcal{N}$, respectively, at each timestep of an experiment. The credo model only considers fixed reward-sharing parameters for each group (self, teams, system). Furthermore, a limitation of the credo model is that it specifies identical, predefined reward-sharing parameters for all agents.

Appropriate social structures and credo parameters can significantly improve how agents learn by facilitating the discovery of valuable yet unrewarded behaviors (RCSA pairs) while mitigating the credit assignment challenges posed by larger teams and fully-cooperative populations (Arjona-Medina et al., 2019; Agarwal et al., 2019; Radke et al., 2023a;b). However, these benefits require determining the best fixed credo parameters for a given population structure through extensive parameter sweeps. We address this limitation by enabling agents to dynamically update their individual reward-sharing credo parameters online, thereby learning to balance these conflicting factors autonomously.



(a) Tuning agent framework.

(b) High-level meta-environment.

Figure 1: (a) Our adaptive *tuning agent* framework where each independent learning agent comprises two policies operating on different timescales. (b) Our high-level meta-environment that the credo-policy operates within. Each tuning agent has its own instance of this meta-environment.

4 Adaptive Reward Sharing Framework

We propose an adaptive framework that overcomes limitations of the credo model by allowing agents to learn reward-sharing parameters using RL. Inspired by meta-learning, our approach imbues each agent with a secondary policy that updates its reward-sharing scheme online, thereby shaping the reward function for its primary behavioral policy. We also introduce new credo reward functions to support heterogeneous credo parameters across the population.

Our proposed *tuning agent* architecture is illustrated in Figure 1a. Each agent i is composed of two internal policies, $i = \langle \pi_i, \pi_i^{\mathbf{cr}} \rangle$, operating at different timescales. Agent i's low-level **behavioral policy** (purple solid box), denoted π_i, operates at every timestep within a multiagent environment, observing states and taking actions in the context of other individual learning agents. Agent i's high-level **credo policy** (orange dashed box), denoted $\pi_i^{\mathbf{cr}}$, operates at a longer timescale in a meta-environment, observing how its behavioral policy's reward is shared among different groups and taking actions to update the credo parameters of the agent's reward function. To reduce nonstationarity for the behavioral policy, the credo policy updates at a defined cadence of E episodes (a hyperparameter) rather than within an episode of learning, akin to the psychological processes of skill acquisition (Anderson, 1982). Through its credo policy, each agent adaptively shapes its own reward function to maximize long-term reward within the given social structure. In the mixed-motive environments with RCSA pairs that we consider, maximizing reward requires effective coordination and teamwork between agents.

Since each agent's credo policy updates its own parameters, agents naturally develop distinct reward-sharing configurations, even within the same group. For example, if agent i and agent j share a team, agent i might adopt team-based credo parameter $\phi_i = 0.2$ while agent j settles on $\phi_j = 0.6$, meaning they contribute differently to their shared team-based reward. Allowing for these heterogeneous credo parameter distributions between agents, we define the team-based reward channel for any team T_n as $TR_i^{T_n}$ and define the system-based reward channel as SR_i, given by:

$$TR_i^{T_n} = \sum_{j \in T_n} \phi_j^{T_n} R_j(S, A_j, S) \quad \text{and} \quad SR_i = \sum_{j \in \mathcal{N}} \omega_j R_j(S, A_j, S).$$

Additionally, let IR_i represent the agent's normal individual exogenous reward (i.e., R_i) they receive from the environment for observing the state and taking individual actions. Building on these definitions, each agent's credo-based reward function $R_i^{\mathbf{cr}}$, which its behavioral policy π_i seeks to maximize, is defined as:

$$R_i^{\mathbf{cr}} = \psi_i \cdot IR_i + \sum_{T_n \in \mathcal{T}_i} \frac{\phi_i^{T_n}}{\sum_{j \in T_n} \phi_j^{T_n}} \cdot TR_i^{T_n} + \frac{\omega_i}{\sum_{j \in \mathcal{N}} \omega_j} \cdot SR_i. \tag{1}$$

An agent's credo policy, $\pi_i^{\mathbf{cr}}$, learns to optimize the mean credo-based reward that its behavioral policy accumulates over the E episodes between credo updates. With our credo-based reward, the team and system-based reward channels depend on both the reward and credo parameters of all agents on that respective team or in the system. Since team and system-based rewards are allocated among agents in proportion to their credo parameters, we maintain budget balance: all rewards earned from the environment are fully distributed, with no loss or introduction of excess reward.

Allowing agents to update their reward-sharing parameters introduces a second-order social dilemma that affects both credo and behavioral policies, similar to reward gifting discussed in Section 3. Given the existence of RCSA pairs, agents will extract different amounts of exogenous reward from the environment. Agents that collect more reward from the environment have the short-term incentive to decrease their team and system credo parameters to keep more reward for themselves. However, decreasing these parameters inevitably increases the incentive for their teammates or peers to stop executing RCSA pairs in search of exogenous reward themselves. The decrease in executed RCSA pairs would result in a decrease in rewards able to be obtained from the environment, ultimately decreasing long-term global welfare. While RL has been shown to perform poorly in social dilemmas, our results show that our framework balances these two dilemmas by developing and maintaining effective reward-sharing credo policies and cooperative behavioral policies.

5 Evaluation Details

We outline the setup for evaluating our reward-sharing framework. First, we present the behavioral environments and specify the low-level policy implementation. Next, we introduce the meta-environment and detail how agents adjust their reward functions via a credo policy. Finally, we define our experimental configurations that form the basis of our empirical analysis.

Behavioral Environments and Behavioral Policy Implementation. We utilize two commonly studied stochastic game environments with distinct multiagent challenges as our low-level behavioral environments: the Cleanup Gridworld Game (Cleanup) (Vinitsky et al., 2019) and Neural MMO (NMMO) (Suarez et al., 2019). The different incentives and reward structures of these two environments allows us to evaluate our approach under different environmental conditions with various RCSA pair dynamics. We implement behavioral policies in these environments with Proximal Policy Optimization (PPO) given its effectiveness in multiagent settings (Schulman et al., 2017; Yu et al., 2022; Radke et al., 2023b). All learning trials in Cleanup consist of 3.4×10^8 environmental timesteps and trials in NMMO are 2.6×10^7 environmental timesteps. Episodes in both last for 1,000 timesteps. Agent i's behavioral policy, π_i, aims to maximize its individual credo-based reward function, $R_i^{\mathbf{cr}}$ (Equation 1), which is calculated at each timestep according to the agent's current credo parameters, $\mathbf{cr}_i$.

We define agents to belong to only one team to study the base case scenario. In Cleanup, we define three disjoint teams of two agents each from a population of six agents (i.e., $|\mathcal{T}| = 3$ disjoint teams from the population of $N = 6$ agents where for every agent i, $|T_n| = 2$). In NMMO, we set two disjoint teams of three agents each from the population of six agents (i.e., $N = 6$ and two teams $|\mathcal{T}| = 2$ of three agents $|T_n| = 3$ each). These configurations are consistent with prior work studying teams in both environments (Radke et al., 2023a;b).

Meta-Environment and Credo Policy Implementation. Our tuning agents are comprised of an additional policy that operates in a meta-environment to update parameters of their individual reward functions. Each tuning agent in the behavioral environments has its own instance of this meta-environment that its credo policy is optimizing within. Since agents belong only to one

Table 1: Experiment configurations. Items labeled (1) to (3) correspond to baseline experiment configurations. The bold configurations represent experiments utilizing our tuning agents (4) and learning from learned heterogeneous credo parameter distributions (5).

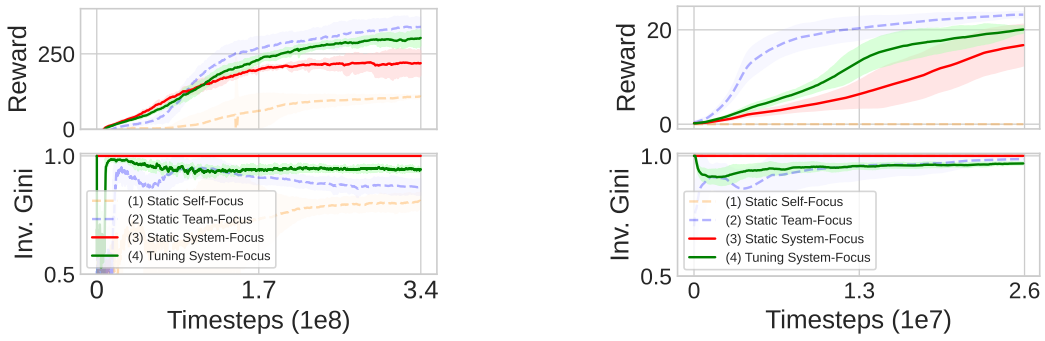
Exp. Configuration	Initial Credo $\langle \psi, \phi, \omega \rangle$	Credo Changes During Exp.	Agent Credo Heterogeneity
(1) Static Self-Focus	$\langle 1, 0, 0 \rangle$	✗	✗
(2) Static Team-Focus	$\langle 0, 1, 0 \rangle$	✗	✗
(3) Static System-Focus	$\langle 0, 0, 1 \rangle$	✗	✗
(4) Tuning System-Focus	$\langle 0, 0, 1 \rangle$	✓	✓
(5) Static Post-Tuning	Learned during (4)	✗	✓

team, the credo policy of an agent learning to modify three parameters (i.e., agent i's credo policy modifies $\mathbf{cr}_i = \langle \psi_i, \phi_i, \omega_i \rangle$). As a proof of concept for our approach, we define the meta-environment (shown in Figure 1b) with discrete credo parameters along intervals of 0.2, creating 21 possible states. The credo policy, $\pi_i^{\mathbf{cr}}$, for each agent is implemented with Q-Learning with ϵ-greedy exploration (Watkins & Dayan, 1992). A credo policy can choose from seven actions, increasing/decreasing any two credo parameters (six actions) or no operation (one action). Infeasible actions are treated as no operation. An agent's credo policy learns a reward-sharing configuration that maximizes the mean credo-based reward that their behavioral policy collects throughout the E episodes since the last credo update. For the following $E \geq 1$ episodes, the behavioral policy optimizes the agent's reward function that is calculated using these updated reward-sharing parameters, $\mathbf{cr}_i'$. Further environment and implementation details are in the Supplementary Materials 8.1.

Experiment Configurations. Table 1 provides an overview of our experiment configurations, which are either *static* or *tuning* scenarios. In **static scenarios** (configurations (1) through (3)), agents' credo parameters are initialized to some predefined *focus* at the start of an experiment and never change during the experiment to serve as comparison baselines. For example, in the *static self-focused* scenario agents are fully selfish and do not change their homogeneous reward-sharing configurations during learning. Static scenario agents are comprised only of a behavioral policy and operate exclusively in the behavioral environments using PPO.

In contrast, our **tuning scenario** consists of agents utilizing our adaptive framework (Figure 1a) being comprised of both a behavioral policy and a credo policy. In the *tuning system-focus* scenario (configuration (4)), credo parameters are initialized to system-focused (i.e., fully cooperative) and are dynamically updated according to our tuning framework. Although prior work has argued that best overall outcomes occur when agents act fully cooperatively (Gemp et al., 2022; Wang et al., 2019), more recent studies suggest that purely cooperative approaches can be suboptimal (Durugkar et al., 2020; Radke et al., 2022; Roesch et al., 2024). So, we adopt a fully cooperative configuration as the initial condition for our agents, since it serves as a natural starting point from which agents should, in theory, do well if they successfully coordinate, but where we know there is room for improvement. In all tuning scenario trials, we define $E = 96$ (96,000 environment timesteps) for the credo policy updates to coincide with our parallel hardware configuration. For all experiments, both static and tuning of all configurations, we perform seven trials (i.e., seeds) and report results with 95% confidence intervals (CI).

In our final experiment, the **static post-tuning** scenario (configuration (5)), we reuse the heterogeneous credo parameters learned by agents during the tuning system-focus trials (configuration (4)) and apply them as static parameters for newly instantiated agents. Specifically, we take the mean credo parameters for each agent from the final 20% of timesteps in each tuning trail, by which point the parameters had stabilized in all of our trials, and designate these as *learned* credo parameters for a new population. This is done for all seven sets of learned credo parameters from the tuning trials, where each set consists of six heterogeneous reward-sharing parameterizations (i.e., one for each of the six agents). Further details on our experiment configurations are in Supplementary Material 8.2.



(a) Cleanup Results.

(b) NMMO Results.

Figure 2: Comparing tuning (solid green) and static (solid red) agents initialized with system-focus credo for mean population reward (top) and inverse Gini index (bottom). For comparison, we gather results for a static team-focused population (dashed blue), the best observed configuration from past work, and results for a static self-focused population (dashed orange) as a baseline. All metrics are from seven trials (mean ± 95% CI).

6 Results

We empirically evaluate the following research questions. **RQ1 in Section 6.1:** How does the performance, in terms of reward and equality, of a population our tuning agents contrast with the performance of a population of static (non-tuning) fully-cooperative agents? **RQ2 in Section 6.2:** How do agents' policies and credo parameters evolve throughout the learning process using our framework? **RQ3 in Section 6.3:** To what extent do the final credo parameter distributions that our tuning agents develop represent effective reward-sharing mixtures for learning and coordination?

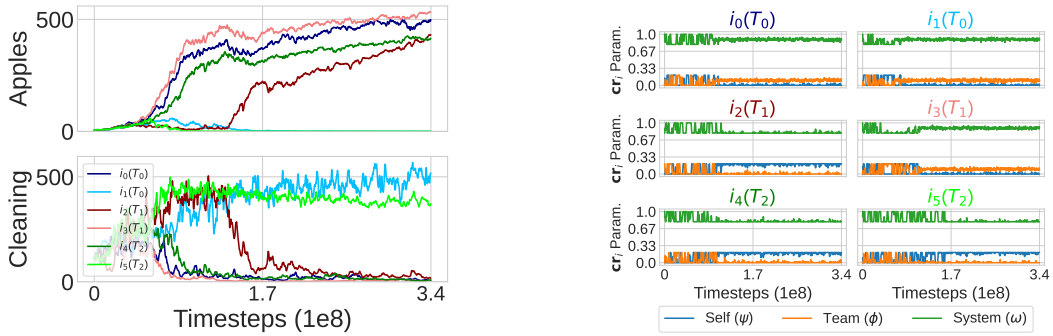
6.1 Tuning Reward-Sharing Parameters

First, addressing RQ1, we compare our tuning agents with several static homogeneous reward sharing baselines. Specifically, we compare mean population reward and inverse Gini index results from tuning system-focus trials (4) with the baseline static configurations of static self-focus (1), static team-focus (2), and static system-focus (3). Figures 2a and 2b illustrate the performance across scenarios in Cleanup and NMMO. The top plot in each figure shows mean population reward, while the bottom plot shows the inverse Gini index, demonstrating equality. All metrics are the mean result across seven trials with 95% confidence intervals.

Our results show that our tuning approach has learning advantages, enabling agents to achieve more reward than their corresponding static agents in both environments. In Cleanup (Figure 2a), tuning system-focus agents (green) achieve 34.2% higher mean population reward on average than the static system-focus population (red) over the last 25% of timesteps. In NMMO (Figure 2b), our results highlight the learning speed benefits that tuning agents exhibit over static agents. For example, tuning system-focus credo agents (green) reach a reward of 10 in 31% fewer timesteps compared to the static system-focus agents (red) while also obtaining 20.3% more reward over the final 25% of experiment timesteps.

As shown by the inverse Gini index results, static system-focus maintains perfect equality by definition as it is fully cooperative. In comparison, the tuning agent population has reduced equality which is a requirement of updating reward the sharing scheme. However, high reward equality is still obtained by tuning agents across both environments, either more or on-par with the static team-focused setting which is the best observed configuration from past work (Radke et al., 2023a).

Our results show the ability for tuning agents to autonomously overcome the learning challenges of a fully cooperative environment and the possibility to obtain significantly more reward than the static



(a) *Cleanup:* Agent behaviors.

(b) *Cleanup:* Agent credo parameters.

Figure 3: Agent behavior and credo parameters throughout one tuning system-focus trial in Cleanup. The six agents are denoted as i_0 though i_5. The three teams are denoted as T_0 through T_2. Agents on the same team share varying shades of the same color in the plot labels. Figure 3a shows the number of apples consumed and the number of river cleaning actions for each agent. Figure 3b illustrates the evolution in self, team, and system credo parameters for each agent, using a sliding window mean across every 10 samples.

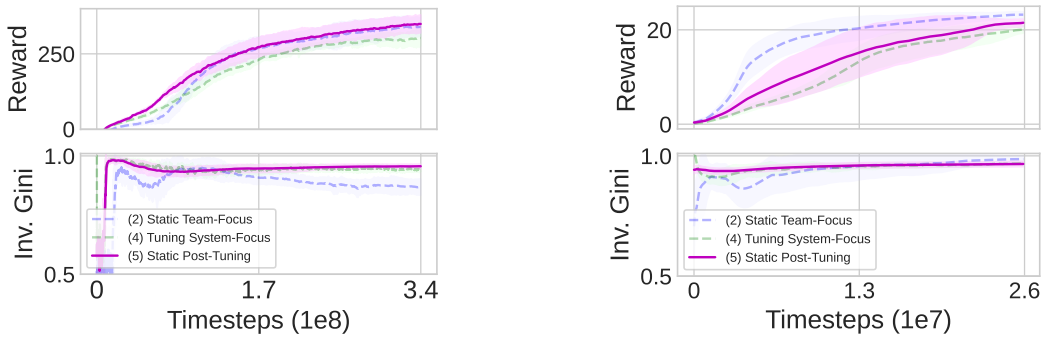
fully cooperative setting. Although the tuning system-focus agents do not achieve the level of reward of the static team-focus agents (blue dashed lines), the tuning process itself still enabled agents to overcome sub-optimal fully-cooperative initializations and yielded effective heterogeneous reward-sharing parameter configurations. We will show that the tuning process develops reward-sharing configurations that can be used as a static reward-sharing initialization which surpasses the reward and equality levels of the static team-focus agents in Cleanup in Section 6.3.

6.2 Learned Behaviors and Learned Reward-Sharing Parameters

Second, addressing RQ2, we examine the impact that the tuning process has on the learned policies of individual agents and their reward-sharing parameters as they evolve throughout an experiment. We illustrate the behavioral policies and credo configurations of agents throughout a tuning system-focus (4) trial to examine the underlying effects of the tuning process on learned behaviors. In NMMO, we do not observe consistent and definitive differences in behaviors between scenarios (static system-focused vs. tuning system-focused). This could be explained by the dynamic nature of the RCSA pairs in NMMO that agents continually adapt towards, resulting in less well defined behavioral *roles* to examine. We confine our analysis in this section to Cleanup.

Learned Behavioral Policies in Cleanup. In Cleanup, agents are rewarded for consuming apples, but apple regrowth rate depends on on the cleanliness of an adjacent river. To be successful in Cleanup, groups must learn to balance rewarded actions of consuming apples with cleaning the river. Figure 3a shows agents' apple consumption (top) and cleaning beam actions (bottom) through one tuning system-focused (4) Cleanup trial (from our results in Figure 2a). Although initialized as system-focused (i.e., fully-cooperative), the tuning agents evolve towards a joint policy that is never learned by any static system-focused population. This advantageous evolution is showcased by agents initial role specialization into three apple-picking and three river-cleaning agents within the first 1.4e8 timesteps, which is the configuration learned by static system-focused agents, but then the tuning agents discover a better joint policy as they continue to learn. Specifically, agent i_2 (dark red) on team T_1 transitions to apple picking in the latter half resulting in the better joint policy of four apple-picking agents and two river-cleaning agents.

This role switching is directly facilitated by our tuning process and the learning of this better joint policy is observed across all seven tuning system-focused trials. While tuning agents do not surpass static team-focused agents in total reward, they converge to the same joint policy of four apple-



(a) Cleanup Results.

(b) NMMO Results.

Figure 4: Examining static agents initialized with learned credo parameter configurations (learned during tuning system-focus trials in Figure 2). We contrast the learned parameter configurations of static post-tuning (solid purple) with tuning system-focus (dashed green) agents, and static team-focus (dashed blue) agents for mean population reward (top) and inverse Gini index (bottom). All metrics are from seven trials (mean ± 95% CI).

picking agents and two river-cleaning agents. This reward differential could be explained by a potential underlying role efficiency advantage corresponding to the increased learning time that an agent population spends executing some joint policy. Thus, our analysis of RQ3 in Section 6.3 explores how the tuning process results in reward-sharing parameter distributions that tend towards better joint policies, and thus higher rewards, when training from scratch.

Learned Credos in Cleanup. Figure 3b show the learned credo parameters for each agent over the course of the tuning system-focus Cleanup trial shown in Figure 3a. Agents began the trial starting with homogeneous fully-cooperative, $cr = \langle 0, 0, 1 \rangle$, reward-sharing parameters and update their credo parameters over the course of the trial using our framework. Our results show that each agent develops a heterogeneous stable credo distribution. We observe that agents i_0 and i_1 (blue; team T_0) adopt some team-focused credo for complementary roles, a river cleaner and an apple picker. Conversely, agents i_4 and i_5 (green; team T_2) adopt similar complementary roles, but maintain more self-focused credo through the experiment. Most notably, agent i_2, in T_1, that switches roles, as mentioned in the previous section, becomes slightly self-focused around 1.4e8 timesteps which coincides with their role switch. A potential contributor to this role switch could be that, because this agent has a more self-focused credo, they keep a larger share of their reward, making it more advantageous for them to take on the role of an apple picker. Altogether, agents discover a better global joint policy while maintaining high reward equality through the process of learning their heterogeneous credo parameter distributions. These results show how our framework allows agents to autonomously learn differing reward-sharing parameters that enhance global outcomes.

6.3 Learning with Learned Reward-Sharing Parameters

Finally, addressing RQ3, we instantiate new static agents with the heterogeneous learned reward-sharing parameters that agents converge to at the end of the tuning trials in configuration (4). These results, called the static post-tuning experiment (5), are compared to tuning system-focused results (4), from which the learned parameters originate, and the previously best observed static-team focus configuration (2). Our findings are illustrated in Figure 4.

In Cleanup (Figure 4a), we find that the static post-tuning agents achieve more reward and higher equality than the tuning system-focus results. Interestingly, while the tuning system-focus experiment outperforms the fully cooperative static system-focused setting (as shown in Figure 2), the tuning process develops a reward-sharing configuration that outperforms itself when used statically and trained with from scratch. Furthermore, the static post-tuning results surpass the previously best

observed result of the static team-focused population, achieving both slightly higher mean population reward and significantly higher equality. In prior work, discovering this effective team-focused configuration for a given social structure required an extensive hyperparameter sweep over the credo parameters (Radke et al., 2023a). With a discretized credo space and homogeneous parameters among agents, this necessitated at least 21 learning trials to explore the space. If extended to heterogeneous parameters, as in our work, the search space grows exponentially, making exhaustive exploration infeasible. By contrast, our approach enables agents to autonomously learn effective reward-sharing parameters for a given social structure, potentially requiring as few as a single learning trial to identify a strong configuration. Thus, our static post-tuning is now the best observed result in Cleanup with teams while being autonomously discovered using RL.

In NMMO (Figure 4b), static post-tuning agents also outperform the results of the tuning system-focus agents in the tail end of the experiment. However, the static post-tuning agents do not outperform the static team-focused population in NMMO. This could be explained by the multiple dynamic RCSA pairs in NMMO which differs from Cleanup, by some other underlying environment dynamic, or social structure dependency. More concretely exploring the relationship between environment dynamics, social structure, and reward sharing may prove to be an interesting line of future work which approaches such as ours could help autonomously disentangle.

7 Conclusion

We introduce a decentralized, multi-level framework that empowers each agent to learn both its behavior and heterogeneous reward-sharing parameters in complex, mixed-motive environments with social structure. By coupling a low-level behavioral policy with a high-level reward-sharing policy, our approach enables agents to adjust their incentives and adapt to the surrounding social context. Letting each agent intelligently refine how their rewards are distributed among the population helps shape the incentive landscape to foster effective coordination.

Our empirical results demonstrate that, starting for a sub-optimal yet commonly used fully cooperative initialization, agents can substantially improve global outcomes by adapting their reward-sharing parameters. In two extensively studied multiagent environments, Cleanup and NMMO, our method outperforms static, non-adaptive fully cooperative baselines. Further, in Cleanup, the heterogeneous configurations learned during online tuning achieve the highest observed results, surpassing all static baselines for reward while maintaining high equality. This capability alleviates the need for exhaustive or heuristic sweeps over potential reward-sharing parameterizations, which become oppressive in heterogeneous settings. However, further work is needed to address the challenges of more complex environments like NMMO in which the learned heterogeneous reward-sharing schemes do not outperform static team-focus populations.

Our framework offers a promising approach towards autonomously balancing challenges related to credit assignment, value discovery, and overall joint policy development in social contexts. While the credo policy is discretized in our current implementation, future work could model different components of agents' reward functions more explicitly, such as through continuous action spaces as in recent deep RL research (e.g., (MacGlashan et al., 2022)). Such an extension would expand reward function decomposition to multiagent settings and increase the richness of the reward-sharing search space. Overall, our findings highlight the potential of adaptive, self-tuning meta approaches to improve the performance of individual learning agents in multiagent systems characterized by diverse social structures and varying incentives.

References

Akshat Agarwal, Sumit Kumar, and Katia Sycara. Learning transferable cooperative behavior in multi-agent teams. *arXiv preprint arXiv:1906.01202*, 2019.

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.

John R Anderson. Acquisition of cognitive skill. *Psychological Review*, 89(4):369, 1982.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

Phillip JK Christoffersen, Andreas A Haupt, and Dylan Hadfield-Menell. Get it in writing: Formal contracts mitigate social dilemmas in multi-agent rl. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 448–456, 2023.

Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R McKee, Joel Z Leibo, Kate Larson, and Thore Graepel. Open problems in cooperative AI. *arXiv preprint arXiv:2012.08630*, 2020.

Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative AI: Machines must learn to find common ground. *Nature*, 593:33–36, 2021.

Heng Dong, Tonghan Wang, Jiayuan Liu, Chi Han, and Chongjie Zhang. Birds of a feather flock together: A close look at cooperation emergence via multi-agent RL. *arXiv preprint arXiv:2104.11455*, 2021.

Ishan Durugkar, Elad Liebman, and Peter Stone. Balancing individual preferences and shared objectives in multiagent reinforcement learning. *Good Systems-Published Research*, 2020.

Ian Gemp, Kevin R McKee, Richard Everett, Edgar A Duéñez-Guzmán, Yoram Bachrach, David Balduzzi, and Andrea Tacchetti. D3C: Reducing the price of anarchy in multi-agent learning. *Proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems*, 2022.

Edward Hughes, Joel Z. Leibo, Matthew Phillips, K. Tuyls, Edgar A. Duéñez-Guzmán, A. Castañeda, Iain Dunning, Tina Zhu, Kevin R. McKee, R. Koster, Heather Roff, and T. Graepel. Inequity aversion improves cooperation in intertemporal social dilemmas. In *NeurIPS*, 2018.

Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 464–473, 2017.

Andrei Lupu and Doina Precup. Gifting in multi-agent reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pp. 789–797, 2020.

James MacGlashan, Evan Archer, Alisa Devlic, Takuma Seno, Craig Sherstan, Peter R Wurman, and Peter Stone. Value function decomposition for iterative design of reinforcement learning agents. *NeurIPS*, 2022.

Kevin R McKee, Ian Gemp, Brian McWilliams, Edgar A Duèñez-Guzmán, Edward Hughes, and Joel Z Leibo. Social diversity and social preferences in mixed-motive reinforcement learning. *AAMAS*, 2020.

David Radke, Kate Larson, and Tim Brecht. Exploring the benefits of teams in multiagent learning. In *IJCAI*, 2022.

David Radke, Kate Larson, and Tim Brecht. The importance of credo in multiagent learning. *Proceedings of the 22nd International Conference on Autonomous Agents and MultiAgent Systems*, 2023a.

David Radke, Kate Larson, Tim Brecht, and Kyle Tilbury. Towards a better understanding of learning with multiagent teams. *IJCAI*, 2023b.

Stefan Roesch, Stefanos Leonardos, and Yali Du. The selfishness level of social dilemmas. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 2441–2443, 2024.

Jürgen Schmidhuber. *Evolutionary Principles in Self-Referential Learning, or On Learning How to Learn: The Meta-Meta-... Hook*. PhD thesis, Technische Universität München, 1987.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, 2017.

Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*, 2019.

The Ray Team. RLlib: Industry-Grade Reinforcement Learning. https://docs.ray.io/en/latest/rllib/index.html, 2023. Accessed 2025-02-20.

Eugene Vinitsky, Natasha Jaques, Joel Leibo, Antonio Castenada, and Edward Hughes. An open source implementation of sequential social dilemma games. https://github.com/eugenevinitsky/sequential_social_dilemma_games/issues/182, 2019. GitHub repository.

J. X. Wang, E. Hughes, C. Fernando, W. M. Czarnecki, E. A. Duéñez-Guzmán, and J. Z. Leibo. Evolving intrinsic motivations for altruistic behavior. In *AAMAS*, pp. 683–692, 2019.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Jiachen Yang, Ang Li, Mehrdad Farajtabar, Peter Sunehag, Edward Hughes, and Hongyuan Zha. Learning to incentivize other learning agents. *Advances in Neural Information Processing Systems*, 33:15208–15219, 2020.

Yuxuan Yi, Ge Li, Yaowei Wang, and Zongqing Lu. Learning to share in multi-agent reinforcement learning. *NIPS '22*, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Supplementary Materials

The following content was not necessarily subject to peer review.

8 Supplementary Materials

8.1 Environment and Implementation Details

In order to evaluate our agent framework, we use two commonly studied stochastic game environments with distinct multiagent challenges, like social dilemmas and resource management, as low-level behavioral environments. Alongside these, we introduce a high-level meta-environment for our framework wherein agents learn to optimize their reward-sharing strategies.

Low-Level Behavioral Environments. We utilize the Cleanup Gridworld Game (Cleanup) (Vinitsky et al., 2019) and Neural MMO (NMMO) (Suarez et al., 2019) as the low-level environments to serve as the primary context in which agents interact and learn. The different incentives and reward structures of these two environments allows us to evaluate our approach under different environmental conditions. Cleanup presents a social dilemma where successful groups of agents must perform RCSA pairs through active provisioning (Hughes et al., 2018). Agents are rewarded for consuming apples, but apple regrowth rate depends on on the cleanliness of an adjacent river. To be successful in Cleanup, groups must learn to balance rewarded actions of consuming apples with cleaning the river. In contrast, NMMO presents a hunter-gatherer structure where successful groups must learn to forage different resources and execute multiple RCSA pairs concurrently. Agents must maintain a stash of consumable resources (food and water) that deplete each timestep but are replenished by harvesting in the environment. Agents are rewarded for positive increases to their lowest resource. Thus, to be successful in NMMO, agents must learn to maintain both food and water to receive reward, creating multiple dynamically changing RCSA pairs.

We implement the behavior policy, π_i, of each agent, as well as all baseline (i.e., non-tuning) agents, in these environments with Proximal Policy Optimization (PPO) given its effectiveness in multiagent settings (Schulman et al., 2017; Yu et al., 2022; Radke et al., 2023b). We provide the full set of PPO hyperparameters in our code repository for transparency and reproducibility. Our PPO hyperparameters are chosen based on values that have been shown to achieve strong performance in prior works. Importantly, we do not tune these hyperparameters between methods, both our approach and the baselines use the same PPO hyperparameter settings to ensure a fair comparison. At every step of an episode, agents select their individual action given their individual partially-observable observation. Agent i's behavioral policy aims to maximize their individual credo-based reward function, $R_i^{\mathbf{cr}}$ (Equation 1), which is calculated at each timestep according to the agent's current credo parameters, $\mathbf{cr}_i$. Our tuning agents are comprised of an additional policy that acts in a meta-environment which we describe next.

High-Level Meta-Environment. Figure 1b illustrates our defined meta-environment that an agents' credo policy takes actions within. We assign agents to belong to only one team to study the base case scenario. This results in the credo policy of an agent learning to modify three parameters (i.e., agent i's credo policy modifies $\mathbf{cr}_i = \langle \psi_i, \phi_i, \omega_i \rangle$). To simplify the learning process, we discretize credo parameters to intervals of 0.2 creating a state space of 21 possible states. Note that each tuning agent in the behavioral environments, Cleanup or NMMO, has its own instance of this meta-environment that its credo policy is optimizing within.

The credo policy, $\pi_i^{\mathbf{cr}}$, for each agent is implemented with Q-Learning with ϵ-greedy exploration (Watkins & Dayan, 1992). We implement a tabular Q-learning agent with an initial exploration rate of $\epsilon_0 = 0.2$, which follows an exponential decay: $\epsilon_t = \epsilon_0 e^{-0.01t}$. The learning rate is adaptive and defined as $\alpha(s, a) = \frac{1}{N(s,a)}$, where $N(s, a)$ is the number of times action a has been taken in state s. The discount factor is set to $\gamma = 0.9$. Action selection follows an ϵ-greedy strategy, where the action with the highest Q-value is chosen with probability $1 - \epsilon$, while a random action

(selected uniformly from all possible actions) is taken with probability ϵ. A credo policy can choose from seven actions, increasing/decreasing any two credo parameters (six actions) or no operation (one action). Infeasible actions, such as making a credo parameter negative, are treated as no operation. For the following $E \geq 1$ episodes, the low-level behavioral policy optimizes the agent's reward function that is calculated using these updated reward-sharing parameters, $\mathbf{cr}'_i$.

We implement agents across both low-level and high-level environments in the RLlib library (The Ray Team, 2023). An experiment in Cleanup consists of 3.4×10^8 environmental timesteps and experiments in NMMO are 2.6×10^7 environmental timesteps. Episodes in both Cleanup and NMMO last for 1,000 timesteps. In Cleanup, we define three disjoint teams of two agents each from a population of six agents (i.e., $|\mathcal{T}| = 3$ disjoint teams from the population of $N = 6$ agents where for every agent i, $|T_n| = 2$). In NMMO, we set two disjoint teams of three agents each from the population of six agents (i.e., $N = 6$ and two teams $|\mathcal{T}| = 2$ of three agents $|T_n| = 3$ each). These configurations are consistent with prior work studying teams in both environments (Radke et al., 2023a;b).

8.2 Evaluation Details

Our evaluation is designed to address the following research questions. **(RQ1)** How does the performance, in terms of reward and equality, of a population of tuning agents contrast with the performance of a population of static (non-tuning) fully cooperative agents? **(RQ2)** How do agents' policies and credo parameters evolve throughout the learning process using our framework? **(RQ3)** To what extent do the final credo parameter distributions that tuning agents develop represent effective reward-sharing mixtures for learning and cooperation? In the remainder of this section, we outline our experimental trial configurations and the specific experiments that we run in order to address our research questions.

Experiment Configurations. Table 1 provides a summary of our experimental scenario configurations and their distinctions for ease of reference. Throughout our experiments, we refer to *static* scenarios and *tuning* scenarios.

In **static scenarios**, agents' credo parameters remain constant after being initialized to a predefined *focus* at the start of an experiment. A *static self-focus* scenario (configuration (1) in Table 1) is where all of an agent's credo is weighted to the self parameter (ψ_i), i.e., $\mathbf{cr}_i = \langle 1.0, 0.0, 0.0 \rangle$. This is equivalent to a fully selfish population wherein no agents share reward. A *static team-focus* scenario (configuration (2) Table 1) is where all of an agent's credo is weighted to the team parameter ($\phi_i^{T_i}$), i.e., $\mathbf{cr}_i = \langle 0.0, 1.0, 0.0 \rangle$, and only teammates share rewards equally. Finally, *static system-focus* scenario (configuration (3) in Table 1) is where all of an agent's credo value is placed on the system parameter (ω_i), i.e., $\mathbf{cr}_i = \langle 0.0, 0.0, 1.0 \rangle$. A system-focus scenario is equivalent to a fully cooperative population since all agents equally share rewards within the system. Agent populations in static scenarios typically have homogeneous credo parameters, meaning that all agents in the population have the same reward-sharing parameters. A notable exception to this, which we detail later in this section, is the *static post-tuning* configuration wherein agents have heterogeneous credo parameters. In static scenarios, agents are only comprised of a behavioral policy, π_i, and only act in the low-level behavioral environments.

In contrast, **tuning scenarios** consist of agents utilizing our framework, being comprised of both a behavioral policy and a credo policy, $\pi_i^{\mathbf{cr}}$ (recall Figure 1a). In our *tuning system-focus* scenario (configuration (4) in Table 1), credo parameters are initialized to system-focused (fully cooperative) and are dynamically updated according to our tuning framework. In all tuning scenario trials, we define $E = 96$ (96,000 environment timesteps) for the credo policy updates to coincide with our parallel hardware configuration. For all experiments, both static and tuning of all configurations, we perform seven trials to reduce variability and report results with 95% confidence intervals.

In our *static post-tuning* scenario (configuration (5) in Table 1) we use the credo parameters learned by agents during the tuning system-focus trials (configuration (4)) and apply them as static parame-

ters for newly instantiated agents. Specifically, we take the average credo parameters for each agent from the final 20% of timesteps in each tuning trail, by which point the parameters had stabilized in all of our trials, and designate these as *learned* credo parameters. These learned credo parameters are then used as static reward-sharing configurations, meaning the agents' reward-sharing parameters remain constant throughout learning, for our static post-tuning scenario. For each static post-tuning trial, we initialize the new agents with the corresponding learned credo parameters from one of the previous seven tuning trials. This is done for all seven sets of learned credo parameters from the tuning experiments, where each set consists of six reward-sharing parameterizations, one for each of the six agents.

Experiments. The experiments we conduct are as follows. First, we initialize agents to be fully-cooperative and subsequently update their reward-sharing parameters using our tuning framework in order to determine its effects on agent outcomes. (RQ1 addressed in Section 6.1). Specifically, we compare mean population reward and inverse Gini index results from tuning system-focus trials (4) with the baseline static configurations of static self-focus (1), static team-focus (2), and static system-focus (3). Past work has shown that a fully cooperative population (i.e., static system-focus (3)) achieves sub-optimal results in these environments (Radke et al., 2023a;b). Thus, our tuning agents must overcome this sub-optimal learning environment where credit assignment is challenging to maximize performance. Second, we examine the impact that the tuning process has on the learned policies of individual agents and their credo configurations as they evolve throughout an experiment (RQ2 addressed in 6.2). We illustrate the behavioral policies and credo configurations of agents throughout a tuning system-focus (4) trial to examine the underlying effects of the tuning process on learned behaviors. Finally, we instantiate new static agents with the learned credo parameters that agents converge to at the end of the tuning experiments in configuration (4) (RQ3 addressed in 6.3). In particular, we compare results from static post-tuning trials (5) with tuning system-focus (4) and static team-focus (2) in order to investigate if the agent population discovers favorable reward-sharing configurations during the tuning process that can be used in subsequent learning trials to enable more productive and equitable populations.