Centralized Adaptive Sampling for Reliable Co-Training of Independent Multi-Agent Policies

Nicholas E. Corrado, Josiah P. Hanna

Keywords: multi-agent reinforcement learning, policy gradient, adaptive sampling, on-policy

Summary

Independent on-policy policy gradient algorithms are widely used for multi-agent reinforcement learning (MARL) in cooperative and no-conflict games, but they are known to converge sub-optimally when each agent's individual policy gradient points away from an optimal joint equilibrium (Claus & Boutilier, 1998; Lyu et al., 2021; Papoudakis et al., 2020b; Christianos et al., 2022). Going beyond prior work, we observe that sub-optimal convergence can still arise *even when the expected individual policy gradients of each agent point toward the optimal joint solution.* After collecting a finite set of trajectories, stochasticity in independent action sampling can cause the joint data distribution to deviate from the expected joint on-policy distribution. This *sampling error* w.r.t. the joint on-policy distribution produces inaccurate gradient estimates that can make agents converge sub-optimally. In this work, we show that joint sampling error can be reduced through coordinated action selection and that reducing joint sampling error improves the *reliability* of on-policy policy gradient MARL (*i.e.* the probability of agents converging to an optimal joint policy).

Contribution(s)

1. Our work is the first to show that joint sampling error can cause sub-optimal convergence in independent on-policy policy gradient RL *even when the objective's expected gradient points toward optimal behavior.*
   Context: Most prior works focus on mitigating sub-optimal convergence via equilibrium selection: if optimizing the expected return yields sub-optimal equilibria, the objective is modified to steer agents toward more favorable equilibria (*e.g.*, Christianos et al. (2022)).
2. We show that coordinated action selection can reduce joint sampling error more sample-efficiently than independent on-policy sampling and MA-PROPS, a multi-agent extension of the PROPS sampling error correction method for single-agent RL (Corrado & Hanna, 2023).
   Context: To test this hypothesis, we introduce an adaptive action sampling approach to reduce joint sampling error in the Centralized Training with Decentralized Execution setting. Our method, **Co**operative **S**ampling **E**rror **R**eduction (CoSER), continually adapts a centralized behavior policy to place higher probability on joint actions that are under-sampled w.r.t. the current joint policy. CoSER is algorithmically similar to MA-PROPS but addresses a different problem: MA-PROPS targets sampling error w.r.t. *each agent's marginal action distribution*, whereas CoSER targets *joint* sampling error.
3. We empirically demonstrate that CoSER requires fewer samples than on-policy sampling and MA-PROPS to achieve the same reduction in joint sampling error and also increases the probability of on-policy policy gradient MARL converging optimally.
   Context: With fixed agent policies, CoSER reduces joint sampling error by the same amount as on-policy sampling and MA-PROPS with 30%–50% fewer samples (Fig. 4a). In fact, MA-PROPS generally does not reduce joint sampling error. During RL training, CoSER reduces joint sampling error more than on-policy sampling and MA-PROPS (Fig. 6), thereby increasing the probability of converging optimally by 10%–20% (Fig. 5). MA-PROPS does not improve reliability in nearly all games we study.

Centralized Adaptive Sampling for Reliable Co-Training of Independent Multi-Agent Policies

Nicholas E. Corrado[1], **Josiah P. Hanna**[1]
ncorrado@wisc.edu, jphanna@cs.wisc.edu

[1]**University of Wisconsin – Madison**

Abstract

Independent on-policy policy gradient algorithms are widely used for multi-agent reinforcement learning (MARL) in cooperative and no-conflict games, but they are known to converge sub-optimally when each agent's individual policy gradient points away from an optimal joint equilibrium (Claus & Boutilier, 1998; Lyu et al., 2021; Papoudakis et al., 2020b; Christianos et al., 2022). Going beyond prior work, we observe that sub-optimal convergence can still arise *even when the expected individual policy gradients of each agent point toward the optimal joint solution.* After collecting a finite set of trajectories, stochasticity in independent action sampling can cause the joint data distribution to deviate from the expected joint on-policy distribution. This *sampling error* w.r.t. the joint on-policy distribution produces inaccurate gradient estimates that can make agents converge sub-optimally. We hypothesize that joint sampling error can be reduced through coordinated action selection and that doing so will increase the *reliability* of policy gradient learning in MARL (*i.e.*, the probability of converging to an optimal joint policy). To test this hypothesis, we first introduce an adaptive action sampling approach to reduce joint sampling error in the Centralized Training with Decentralized Execution setting. Our method, **Co**operative **S**ampling **E**rror **R**eduction (CoSER), continually adapts a centralized behavior policy to place higher probability on joint actions that are under-sampled w.r.t. the current joint policy. We then empirically evaluate CoSER on a diverse set of games and demonstrate that (1) CoSER reduces joint sampling error more sample-efficiently than independent on-policy sampling and (2) this reduction increases the reliability of on-policy policy gradient algorithms.

1 Introduction

On-policy policy gradient methods are among the most popular algorithms for multi-agent reinforcement learning (MARL) in cooperative and no-conflict games (De Witt et al., 2020; Yu et al., 2022; Zhou et al., 2021).[1] A common approach is to treat agents as independent learners: each agent samples trajectories independently from its own policy (without observing the actions of other agents), estimates a Monte Carlo approximation of its policy gradient (Sutton & Barto, 2018), and then performs a local policy update to maximize its expected return via gradient ascent. While independent algorithms have demonstrated strong empirical performance (Papoudakis et al., 2020b;a) and powered several high-profile success stories (De Witt et al., 2020; Yu et al., 2022; Zhou et al., 2021), it is well-understood that they may not converge to the most preferred equilibrium even in no-conflict games (Claus & Boutilier, 1998; Lyu et al., 2021; Papoudakis et al., 2020b; Christianos et al., 2022). In this paper, we identify a novel failure mode for independent policy gradient learning in MARL: *even when the expected policy gradients of each agent align with optimal behavior*,

[1]In no-conflict games, all agents share the same set of preferred equilibria. Cooperative games are a subset of no-conflict games in which all agents share the same reward function.

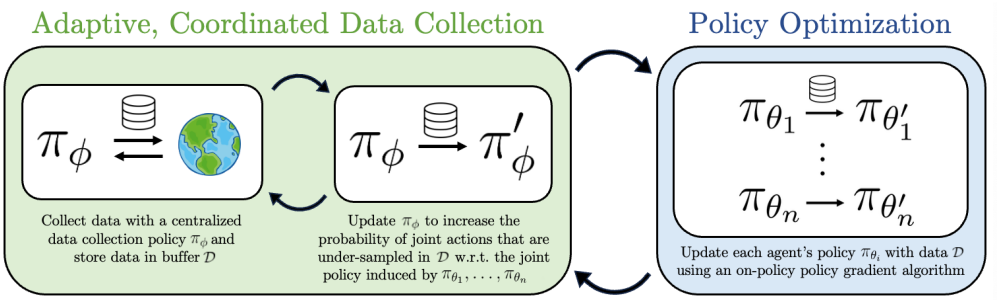


Figure 2: Rather than collecting data $\mathcal{D}$ by sampling actions from each agent's current policy $\pi_{\boldsymbol{\theta}_i}$, CoSER collects data with a centralized policy $\pi_{\boldsymbol{\phi}}$ that it continually adapts to reduce joint sampling error in $\mathcal{D}$ with respect to the joint policy.

stochasticity in action sampling can nevertheless cause agents to converge to sub-optimal solutions. We illustrate this phenomenon with the following example.

Consider the 2×2 matrix game in Fig. 1. Suppose each agent assigns equal probability to actions A and B and selects actions independently. For both agents, the expected reward of A is $(12 + 0)/2 = 6$, and the expected reward of B is $(6 + 2)/2 = 4$. Thus, both agents are incentivized to increase the probability of A, steering them toward the optimal joint action (A, A). Now suppose the agents play the game four times. In expectation, each agent will sample both actions twice and observe each joint action once. However, due to randomness in action selection, Agent 1 may sample A, B, A, B while Agent 2 samples B, A, B, A so that (A, A) and (B, B) are *under-sampled* w.r.t. the expected joint on-policy distribution. Consequently, both agents observe reward 0 for action A and reward 6 for action B and thus increase the probability of B, driving agents toward the sub-optimal strategy (B, B). The core issue is *joint sampling error*: randomness in action sampling causes the empirical joint data distribution to deviate from the expected joint on-policy distribution, leading to inaccurate policy gradient estimates. Moreover, joint sampling error exists even though both agents sample A and B twice and thus have zero sampling error w.r.t. their own policies.

		Agent 2	
		A	B
Agent 1	A	$12, 12$	$0, 6$
	B	$6, 0$	$2, 2$

Figure 1: Matrix game where r_1, r_2 denotes rewards for Agents 1 and 2, respectively.

Under on-policy sampling, the only way to reduce sampling error is to collect more data. Recently, Corrado & Hanna (2023) introduced an action sampling algorithm (PROPS) that reduces sampling error more sample-efficiently than on-policy sampling. However, this work focused on reducing sampling error w.r.t. a single policy in single-agent RL settings. As the above example shows, reducing sampling error w.r.t. each agent's policy individually does not necessarily reduce sampling error w.r.t. the joint policy. Thus, agents must coordinate action selection to reduce joint sampling error. These observations motivate the central question of this work: *Can adaptive, coordinated action sampling reduce joint sampling error and, in doing so, increase the reliability of multi-agent policy gradient RL?* Here, reliability refers to the probability of converging to an optimal joint policy.

To answer this question, we introduce **Co**operative **S**ampling **E**rror **R**eduction (CoSER), an action sampling algorithm that adaptively corrects joint sampling error during multi-agent on-policy data collection (Fig. 2). Rather than sampling actions from each agent independently, CoSER samples actions from a centralized data collection policy that we continually update to increase the probability of under-sampled joint actions. We first evaluate CoSER on a diverse set of no-conflict multi-agent games and show that it reduces joint sampling error more sample-efficiently than on-policy sampling. Next, we answer our central question affirmatively and show that reducing joint sampling error during on-policy policy gradient learning increases the fraction of training runs that converge to an optimal joint policy. While centralized action sampling may be challenging with many agents due to exponential growth of the joint action space, CoSER is a first step toward understanding and mitigating joint sampling error in MARL. In summary, our contributions are:

1. We identify a novel failure mode of independent on-policy policy gradient algorithms: even when the expected gradient points toward optimal behavior, joint sampling error can cause convergence to sub-optimal solutions.
2. We show how this failure mode can be mitigated by coordinating action selection to reduce joint sampling error. To evaluate this idea empirically, we develop CoSER, a data collection algorithm that adaptively increases the probability of under-sampled joint actions.
3. We empirically demonstrate that CoSER reduces joint sampling error more sample-efficiently than on-policy sampling and consequently makes on-policy policy gradient MARL converge to an optimal joint policy more reliably.

2 Related Work

Relative overgeneralization and equilibrium selection. MARL algorithms may converge to sub-optimal solutions due to relative overgeneralization (RO), a phenomenon in which agents overfit to suboptimal behavior of other agents (Wiegand, 2004; Wei & Luke, 2016). The most common cause of RO is *action shadowing* (Fulda & Ventura, 2007): an agent's individually optimal action is suboptimal for the joint policy, causing the expected gradient to point toward a sub-optimal, *risk-dominant* equilibrium (Albrecht et al., 2024). Prior works address equilibrium selection by learning joint action-values (Christianos et al., 2022; Littman et al., 2001; Sunehag et al., 2017; Rashid et al., 2020; Son et al., 2019) or by using optimism (Matignon et al., 2007; Palmer et al., 2017; Zhao et al., 2023). Prior works modify the learning objective because directly optimizing the expected return can lead to sub-optimal equilibria. In contrast, we study settings where the expected gradients already point toward the optimal equilibrium and are the first to show how RO can arise from joint sampling error alone. Rather than altering the objective or gradient update, we improve reliability by reducing joint sampling error through adaptive data collection.

Reducing sampling error via adaptive data collection. Prior work in single-agent RL has shown that adaptive action sampling can reduce sampling error more sample-efficiently than standard on-policy sampling. Zhong et al. (2022) first demonstrated this idea theoretically, and Corrado & Hanna (2023) introduced a practical and scalable algorithm (PROPS) for applying it in policy gradient learning. Mukherjee et al. (2022) uses adaptive sampling in the bandit setting, and other bandits works (Tucker & Joachims, 2022; Wan et al., 2022; Konyushova et al., 2021) use data-conditioned but non-adaptive sampling strategies. These methods focus on single-agent RL or policy evaluation. In contrast, we focus on multi-agent settings and highlight unique challenges posed by joint sampling error.

Reducing sampling error via importance sampling. Several works use importance sampling to reduce sampling error without collecting additional data for policy evaluation (Precup, 2000), policy gradient RL (Papini et al., 2018; Metelli et al., 2018; Hanna et al., 2021), and temporal difference learning (Pavse et al., 2020). Similar techniques exist for contextual bandits (Li et al., 2015; Narita et al., 2019). These works reduce sampling error by reweighting previously collected data. In contrast, our work focuses on whether sampling error can be controlled as data is collected.

Coordinated Exploration. In MARL, data collection techniques often guide action selection to maximize state-action coverage and explore states where multi-agent interactions are likely (Mahajan et al., 2019; Wang et al., 2019; Liu et al., 2021; Li et al., 2022; Gupta et al., 2021; Zheng et al., 2021; Zhang et al., 2023). In contrast, our work adapts action selection to more accurately approximate the on-policy distribution and improve policy gradient estimates.

3 Preliminaries

In this section, we formalize the multi-agent RL setting and discuss relevant multi-agent policy gradient algorithms.

3.1 Multi-Agent Reinforcement Learning

We model the MARL environment as a fully observable, finite-horizon stochastic game $(\mathcal{I}, \mathcal{S}, \mathcal{A}, p, r, \gamma)$ with n agents $\mathcal{I} = \{1, \ldots, n\}$, joint state space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$, and joint action space $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$. Each agent i has a discrete action space $\mathcal{A}_i = \{1, \ldots, k\}$ and a stochastic policy $\pi_{\boldsymbol{\theta}_i} : \mathcal{S}_i \times \mathcal{A}_i \to [0, 1]$ parameterized by $\boldsymbol{\theta}_i$. The joint policy $\pi_{\boldsymbol{\theta}} = (\pi_{\boldsymbol{\theta}_1}, \ldots, \pi_{\boldsymbol{\theta}_n})$ with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ defines a distribution over joint actions conditioned on the joint state. For brevity, we often write $\pi_i := \pi_{\boldsymbol{\theta}_i}$ and $\pi := \pi_{\boldsymbol{\theta}}$. The transition function $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ specifies the probability of transitioning to the next joint state. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$ returns a reward vector $(r_1, \ldots, r_n)$, where r_i is the reward for agent i. Each agent's expected return is $J_i(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{H} \gamma^t r_i(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$, where random variable H is the horizon, and the global objective is to maximize *welfare*, defined as $J(\boldsymbol{\theta}) = \sum_{i \in \mathcal{I}} J_i(\boldsymbol{\theta})$. We focus on *no-conflict games*, where all agents share the same set of optimal joint policies. We refer to the policies used for data collection as *behavior policies* and the policies being optimized as *target policies*.

3.2 On-Policy Policy Gradient Algorithms

On-policy policy gradient algorithms perform gradient ascent over policy parameters to maximize each agent's expected return $J_i(\boldsymbol{\theta})$. For exposition, we assume finite state and action spaces so that the policy gradient w.r.t. agent i can be written as:

$$\nabla J_i(\boldsymbol{\theta}) \propto \sum\nolimits_{(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) \in \mathcal{S} \times \mathcal{A}} d_\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) \left[A_i^\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) \nabla_{\boldsymbol{\theta}_i} \log \pi_i(\boldsymbol{a}_i | \boldsymbol{s}_i) \right] \tag{1}$$

where $\boldsymbol{a}_i$ denotes agent i's action, $\boldsymbol{a}_{-i}$ denotes the actions of all agents except agent i, $d_\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$ denotes the joint state–action visitation distribution, and $A_i^\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$ denotes the advantage of agent i. In practice, $\nabla J_i(\boldsymbol{\theta})$ is approximated with Monte Carlo samples collected from π, and an estimate of A^π is used in place of the true advantage (Schulman et al., 2016). Independent algorithms like IPPO (De Witt et al., 2020) apply single-agent RL algorithms to each agent, treating other agents as part of the environment. Since independent updates may converge to sub-optimal equilibria (Claus & Boutilier, 1998), many algorithms use Centralized Training with Decentralized Execution (CTDE), which gives agents access to global information during training while requiring local execution. CTDE methods typically use centralized critics to improve credit assignment (Foerster et al., 2018; Kuba et al., 2021; Yu et al., 2022; Ma & Luo, 2022; Zhong et al., 2024). We view CTDE methods like MAPPO (Yu et al., 2022) as independent algorithms because their centralized critics depend only on joint observations, not joint actions.

4 Joint Sampling Error in MARL

We now present our primary contribution: a formal description of how joint sampling error produces inaccurate policy gradient estimates and how adaptive sampling can reduce joint sampling error. Observe that the policy gradient of agent i in Eq. 1 is a linear combination of the gradient for each $(\boldsymbol{s}_i, \boldsymbol{a}_i)$ pair $\nabla_{\boldsymbol{\theta}_i} \log \pi_i(\boldsymbol{a}_i | \boldsymbol{s}_i)$ weighted by $d_\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) A^\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$. Crucially, this weighting depends on the *joint* action $(\boldsymbol{a}_i, \boldsymbol{a}_{-i})$. Let $\mathcal{D}$ be a dataset of trajectories. It is straightforward to show that the Monte Carlo estimate of the policy gradient can be written in a similar form as Eq. 1 except with the true state-action visitation distribution replaced with the empirical visitation distribution $d_\mathcal{D}(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$, denoting the fraction of times $(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$ appears in $\mathcal{D}$ (Hanna et al., 2021):

$$\nabla \widehat{J}_i(\boldsymbol{\theta}) = \sum\nolimits_{(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) \in \mathcal{S} \times \mathcal{A}} d_\mathcal{D}(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) \left[A_i^\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) \nabla_{\boldsymbol{\theta}_i} \log \pi_i(\boldsymbol{a}_i | \boldsymbol{s}_i) \right]. \tag{2}$$

By comparing Eq. 1 and Eq. 2, we can see how joint sampling error affects gradient estimation. When $(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$ is over-sampled (*i.e.*, $d_\mathcal{D}(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i}) > d_\pi(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$), then $\nabla_{\boldsymbol{\theta}_i} \log \pi_i(\boldsymbol{a}_i | \boldsymbol{s}_i)$ contributes more to the overall gradient than it should. Similarly, when $(\boldsymbol{s}, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$ is under-sampled, $\nabla_{\boldsymbol{\theta}_i} \log \pi_i(\boldsymbol{a}_i | \boldsymbol{s}_i)$ contributes less than it should. We now provide a concrete example

based on the matrix game in Fig. 1 illustrating how small amounts of joint sampling error can cause the wrong actions to be reinforced—even when agents have access to their true advantages and have zero sampling error in the marginal visitation distribution $d_{\pi_i}(\boldsymbol{s}, \boldsymbol{a}_i)$ of each policy individually.

Example 1: Joint sampling error can cause incorrect policy gradient updates

Consider two policies π_1, π_2 in an MDP with two discrete actions A and B. Assume a direct parameterization $\pi_i(A|\boldsymbol{s}) = \boldsymbol{\theta}_{i,\boldsymbol{s}}$, $\pi_i(B|\boldsymbol{s}) = 1 - \boldsymbol{\theta}_{i,\boldsymbol{s}}$ with $\boldsymbol{\theta}_{i,\boldsymbol{s}_0} = 0.5$ so that each policy places equal probability on both actions in $\boldsymbol{s}_0$ and thus equal probability on each joint action. Then $\forall i$,

$$\nabla \log \pi_i(A|\boldsymbol{s}_0) = -\nabla \log \pi_i(B|\boldsymbol{s}_0), \qquad d_{\pi_i}(\boldsymbol{s}_0, A) = d_{\pi_i}(\boldsymbol{s}_0, B).$$

Suppose that in a particular state $\boldsymbol{s}_0$, the advantages $A_i^\pi(\boldsymbol{s}_0, \boldsymbol{a}_i, \boldsymbol{a}_{-i})$ w.r.t. both policies are $A_i^\pi(\boldsymbol{s}_0, A, A) = 7$; $A_i^\pi(\boldsymbol{s}_0, A, B) = -5$; $A_i^\pi(\boldsymbol{s}_0, B, A) = 1$; $A_i^\pi(\boldsymbol{s}_0, B, B) = -3$, $i = 1, 2$. Then, the expected gradient of π_i increases the probability of sampling A and thus increases the probability of observing the optimal joint action (A, A):

$$2/4 \cdot (7 - 5) \cdot \nabla \log \pi_i(A|\boldsymbol{s}_0) + 2/4 \cdot (1 - 3) \cdot \nabla \log \pi_i(B|\boldsymbol{s}_0) = 2\nabla \log \pi_i(A|\boldsymbol{s}_0) \tag{3}$$

With on-policy sampling, after 4 visits to $\boldsymbol{s}_0$, each joint action will be observed once in expectation. However, if we actually observe (A, B) 2 times and (B, A) 2 times, a Monte Carlo estimate of each policy gradient yields

$$2/4 \cdot (-5) \cdot \nabla \log \pi_i(A|\boldsymbol{s}_0) + 2/4 \cdot (-3) \cdot \nabla \log \pi_i(B|\boldsymbol{s}_0) = -\nabla \log \pi_i(A|\boldsymbol{s}_0) \tag{4}$$

which *decreases* the probability of sampling the optimal (A, A) action. We emphasize that this issue arises despite having zero sampling error w.r.t. the expected on-policy distribution of each policy individually (*i.e.* both agents sample A twice and B twice.)

With on-policy sampling, joint sampling error vanishes only in the limit of infinite data. To more rapidly reduce joint sampling error, agents can instead coordinate their action selection: if a joint action is under-sampled at $\boldsymbol{s}$, agents should increase the probability of sampling that joint action at $\boldsymbol{s}$ in the future. Continuing with Example 1, suppose the agents visit $\boldsymbol{s}_0$ 4 more times. To achieve zero sampling error, the agents should observe (A, A) and (B, B) twice each. Since independent on-policy sampling gives no control over the joint distribution, they may observe (A, B) and (B, A) again. If they sample their next actions from a distribution that puts probability 1 on (A, A) on the first two visits to $\boldsymbol{s}_0$ and probability 1 on (B, B) on the next two, the aggregate batch of data will exactly match the expected joint distribution.

In tabular *single-agent* settings, Zhong et al. (2022) and Corrado & Hanna (2023) proved that selecting the most under-sampled action at each state produces an empirical state-action distribution that converges to expected visitation distribution at a faster rate than on-policy sampling. In multi-agent settings, one might try to apply this heuristic to each agent independently. However, our next example shows that reducing sampling error w.r.t. each agent may not reduce joint sampling error.

Example 2: Independent adaptive sampling may not decrease joint sampling error

Consider two policies π_1 and π_2 that put equal probability on actions A and B so that under on-policy sampling, each joint action is observed equally often in expectation. Now suppose each agent always selects the action most under-sampled relative to its own policy. At $t = 0$, both actions are equally under-sampled, so both agents sample from π_i. Without loss of generality, suppose agent 1 chooses A and agent 2 chooses B. At $t = 1$, agent 1 selects B and agent 2 selects A, since these actions are now under-sampled. This pattern repeats: the agents choose (A, B) at even timesteps and (B, A) at odd timesteps. Consequently, joint actions (A, A) and (B, B) are never sampled, and joint sampling error does not decrease as $t \to \infty$.

This example highlights a key point: correcting joint sampling error requires agents to coordinate their action selection. Building upon these ideas, we now present a *centralized* adaptive sampling algorithm to correct joint sampling error.

5 Reducing Joint Sampling Error

In this section, we introduce an adaptive sampling algorithm to reduce joint sampling error in multi-agent on-policy data collection. Algorithm 1 outlines our CTDE framework in which agents collect data with a centralized behavior policy $\pi_{\boldsymbol{\phi}} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ to train decentralized target policies $\pi_{\boldsymbol{\theta}_i} : \mathcal{S}_i \times \mathcal{A}_i \to [0, 1]$. First, we initialize the behavior policy to be equal to the target policy: $\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s}) = \prod_{i=1}^{n} \pi_{\boldsymbol{\theta}_i}(\boldsymbol{a}_i|\boldsymbol{s}), \forall \boldsymbol{s}$. At each step, agents act according to $\pi_{\boldsymbol{\phi}}$ and add the resulting transition to a buffer $\mathcal{D}$. Every m steps, $\boldsymbol{\phi}$ is updated to increase the probability of under-sampled joint actions in $\mathcal{D}$ w.r.t. $\pi_{\boldsymbol{\theta}}$. Every n steps, each agent updates its policy parameters $\boldsymbol{\theta}_i$ using data from $\mathcal{D}$. To keep the empirical distribution of $\mathcal{D}$ close to the expected joint on-policy distribution, we continually adapt $\pi_{\boldsymbol{\phi}}$ to place more probability on joint actions that are under-sampled w.r.t. $\pi_{\boldsymbol{\theta}}$. Corrado & Hanna (2023) recently introduced an algorithm, PROPS, for making such updates in *single-agent* settings. In what follows, we first review PROPS and describe how its straightforward extension to the multi-agent setting fails to correct joint sampling error. Then, we introduce a new method, CoSER, that uses a centralized behavior policy to reduce joint sampling error.

5.1 Behavior Policy Update: Proximal Robust On-Policy Sampling

PROPS is based on a simple idea: starting at $\boldsymbol{\phi} = \boldsymbol{\theta}$, gradient ascent on the log-likelihood $\mathcal{L}_{\text{LL}}(\boldsymbol{\phi}) = \sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} \log \pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})$ adjusts $\boldsymbol{\phi}$ to match the empirical distribution of $\mathcal{D}$, increasing the probability of over-sampled actions and decreasing that of under-sampled actions. Taking a step in the opposite direction thus increases the probability of under-sampled actions. Zhong et al. (2022) proved that adjusting the behavior policy with a single gradient step in the direction of $-\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\text{LL}}(\boldsymbol{\phi})\big|_{\boldsymbol{\phi}=\boldsymbol{\theta}}$ at each timestep improves the rate at which the empirical data distribution converges to the expected on-policy distribution. To enable larger behavior policy updates that more aggressively correct sampling error, Corrado & Hanna (2023) instead perform gradient ascent on a PPO-inspired clipped surrogate

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{s}, \boldsymbol{a}, \varepsilon) = \min\left[-\frac{\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})}, -\texttt{clip}\left(\frac{\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})}, 1-\varepsilon, 1+\varepsilon\right)\right], \tag{5}$$

where `clip` bounds the ratio $\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})/\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ to the interval $[1-\varepsilon, 1+\varepsilon]$. This objective prevents destructively large updates: it increases the probability of under-sampled actions by at most a factor of $1+\varepsilon$, and decreases that of over-sampled actions by at most $1-\varepsilon$. As in PPO, this clipping enables stable multi-epoch minibatch training while ensuring moderate adjustments to action probabilities.

The natural extension of PROPS to the multi-agent setting uses independent behavior policies $\pi_{\boldsymbol{\phi}_i}(\boldsymbol{a}_i|\boldsymbol{s}_i)$ with the same parameterization as $\pi_{\boldsymbol{\theta}_i}$, yielding a *decentralized* behavior policy $\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s}) = \prod_{i=1}^{n} \pi_{\boldsymbol{\phi}_i}(\boldsymbol{a}_i|\boldsymbol{s}_i)$ where $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$. We call this extension Multi-Agent PROPS (MA-PROPS) and provide pseudocode in Algorithm 2. At each behavior update, we set $\boldsymbol{\phi}_i \leftarrow \boldsymbol{\theta}_i$ and perform minibatch gradient ascent on $\mathcal{L}$. Since $\pi_{\boldsymbol{\phi}}$ is decentralized, MA-PROPS will correct sampling error w.r.t. each agent individually but may not reduce joint sampling error (see Example 2 in Section 4). In the next section, we describe how to correct joint sampling error.

5.2 Cooperative Sampling Error Reduction (CoSER)

To correct sampling error w.r.t. the joint on-policy distribution, the behavior policy $\pi_{\boldsymbol{\phi}}$ must be *centralized*: it must condition on the full joint state $\boldsymbol{s}$ and allow arbitrary dependencies across agents' actions. Moreover, since the PROPS update sets the behavior policy equal to the joint target policy at the start of each behavior update, we also require that $\pi_{\boldsymbol{\phi}}$ can be easily initialized to match $\pi_{\boldsymbol{\theta}}$. Since $\pi_{\boldsymbol{\phi}}$ and $\pi_{\boldsymbol{\theta}}$ have different parameterizations (*i.e.*, $\pi_{\boldsymbol{\theta}}$ is decentralized), we cannot simply set $\boldsymbol{\phi} \leftarrow \boldsymbol{\theta}$. To enable this initialization, we introduce a specialized architecture.

Algorithm 1 On-policy policy gradient algorithm with adaptive sampling

Inputs: Target batch size n, behavior update frequency m
Output: Target policy parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$.
Initialize target policy parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$, initialize behavior policy parameters $\boldsymbol{\phi}$ so that $\pi_{\boldsymbol{\phi}} \equiv \pi_{\boldsymbol{\theta}}$, and initialize empty buffer $\mathcal{D}$.
for timestep $t = 1, 2, \ldots$ **do**
  Collect one transition with $\pi_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s})$, add it to $\mathcal{D}$.
  if $t \mod m \equiv 0$ **then**
    Update $\boldsymbol{\phi}$ with $\mathcal{D}$ using Algorithm 2.
  if $t \mod n \equiv 0$ **then**
    Update $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ with $\mathcal{D}$ using an on-policy policy gradient algorithm.
return $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$

Algorithm 2 Behavior policy update with CoSER or MA-PROPS

Inputs: Joint target policy parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$, buffer $\mathcal{D}$, target KL δ, clipping coefficient ϵ, `n_epoch`, `n_minibatch`.
Output: Behavior policy parameters $\boldsymbol{\phi}$.
if CoSER **then**
  Define *centralized* behavior policy $\pi_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s}) := \sigma(\log \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}) + \Delta_{\boldsymbol{\phi}}(\boldsymbol{s}))$
  Initialize $\boldsymbol{\phi}$ such that $\Delta_{\boldsymbol{\phi}}(\boldsymbol{s}) = \mathbf{0}, \forall \boldsymbol{s} \in \mathcal{S}$ so that $\pi_{\boldsymbol{\phi}} \equiv \pi_{\boldsymbol{\theta}}$
else if MA-PROPS **then**
  Define *decentralized* behavior policy $\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s}) = \prod_{i=1}^{n} \pi_{\boldsymbol{\phi}_i}(\boldsymbol{a}_i|\boldsymbol{s}_i)$.
  Initialize $(\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n) \leftarrow (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ so that $\pi_{\boldsymbol{\phi}} \equiv \pi_{\boldsymbol{\theta}}$
for epoch $i = 1, 2, \ldots,$ `n_epoch` **do**
  for minibatch $j = 1, 2, \ldots,$ `n_minibatch` **do**
    Sample minibatch $\mathcal{D}_j \sim \mathcal{D}$
    Update $\boldsymbol{\phi}$ with a step of gradient ascent on loss $\frac{1}{|\mathcal{D}_j|} \sum_{(\boldsymbol{s},\boldsymbol{a}) \in \mathcal{D}_j} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{s}, \boldsymbol{a}, \varepsilon)$ (Eq. 5)
  if $D_{\text{KL}}(\pi_{\boldsymbol{\theta}}||\pi_{\boldsymbol{\phi}}) > \delta$ **then**
    return $\boldsymbol{\phi}$
return $\boldsymbol{\phi}$

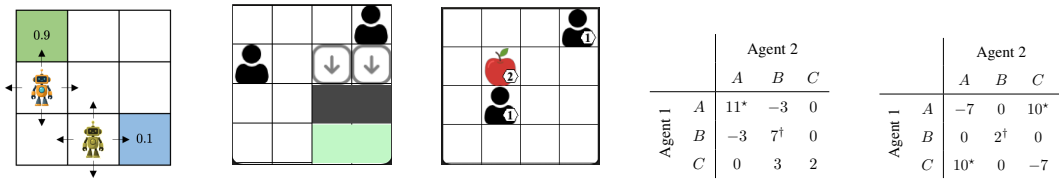
We first compute the logits of the joint policy, $\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \sum_{i=1}^{n} \log \pi_{\boldsymbol{\theta}_i}(\boldsymbol{a}_i|\boldsymbol{s}_i)$. We then define a neural network $\Delta_{\boldsymbol{\phi}}(\boldsymbol{s}) : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$ that outputs an adjustment to each logit. The behavior policy logits are $\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) + \Delta_{\boldsymbol{\phi}}(\boldsymbol{s})$, and the behavior policy is $\pi_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s}) = \sigma\left(\log \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}) + \Delta_{\boldsymbol{\phi}}(\boldsymbol{s})\right)$, where σ denotes the softmax function.[2] To ensure $\pi_{\boldsymbol{\phi}}$ and $\pi_{\boldsymbol{\theta}}$ are equal at the start of each update, we set the final layer of $\Delta_{\boldsymbol{\phi}}$ to the zero vector so that $\Delta_{\boldsymbol{\phi}}(\boldsymbol{s}) = \mathbf{0}$ for all $\boldsymbol{s} \in \mathcal{S}$. Then, we perform minibatch gradient ascent on $\mathcal{L}(\boldsymbol{\phi})$ to adapt $\Delta_{\boldsymbol{\phi}}$, *i.e.*, the logit adjustment $\Delta_{\boldsymbol{\phi}}(\boldsymbol{s})$ will increase for under-sampled joint actions and decrease for over-sampled ones. We call this algorithm **Co**operative **S**ampling **E**rror **R**eduction (CoSER) and provide pseudocode in Algorithm 2.

5.3 CoSER Convergence

In this section, we extend the convergence analysis of Zhong et al. (2022) from the single-agent setting to the multi-agent setting.[3] Due to space constraints, we defer all proofs to Appendix 9. First, we show that CoSER converges to the expected on-policy joint state visitation distribution. Next, we show that under CoSER, the empirical joint policy $\pi_{\mathcal{D}}(\cdot|\boldsymbol{s})$ converges to the joint on-policy distribution $\pi(\cdot|\boldsymbol{s})$ faster than on-policy sampling. We also prove analogous results for each agent individually. Our results use the following assumption:

[2] We omit the behavior policy's dependence on $\boldsymbol{\theta}$ for clarity, as $\boldsymbol{\theta}$ is fixed during behavior updates.

[3] We include this theoretical analysis for completeness and to show how the analysis of Zhong et al. (2022) for the single-agent setting maps to the multi-agent setting. While the second statement of Theorem 2 establishes a novel MARL result, we do not consider this analysis a core contribution of our work.



(a) GridWorld (b) BoulderPush (c) LBF

		Agent 2		
		A	B	C
Agent 1	A	$11^\star$	-3	0
	B	-3	$7^\dagger$	0
	C	0	3	2

(d) Climbing game

		Agent 2		
		A	B	C
Agent 1	A	-7	0	$10^\star$
	B	0	$2^\dagger$	0
	C	$10^\star$	0	-7

(e) Penalty game

Figure 3: A subset of games used in our experiments. In matrix games, $\star$ denotes optimal equilibria, and † denotes suboptimal equilibria. All 2×2 matrix games are listed in Appendix 12.

Assumption 1. *CoSER uses a learning rate of $\alpha \to \infty$ and the behavior policy is parameterized as a softmax function $\pi_\phi(a|s) \propto e^{\phi_{s,a}}$, where each state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ has a parameter $\boldsymbol{\phi}_{s,a}$. This assumption implies that CoSER always selects the most under-sampled joint action in each state.*

We elaborate on why Assumption 1 implies that CoSER always selects the most under-sampled joint action in Appendix 9. Theorem 1 below shows that the empirical state visitation distributions converge to the true state visitation distributions under CoSER. We let $d_{\mathcal{D}_m}$ and $\pi_{\mathcal{D}_m}$ denote the empirical joint state visitation distribution and empirical joint policy after m state-action pairs have been taken, respectively. In particular, $d_{\mathcal{D}_m}(s)$ is the fraction of m joint states that are s, and $\pi_{\mathcal{D}_m}(a|s)$ is the fraction of times that joint action a was observed in joint state s. We use a subscript i to denote analogous quantities for agent i.

Theorem 1. *Assume that $\mathcal{S}$ and $\mathcal{A}$ are finite. Under CoSER with Assumption 1, the empirical joint state visitation distribution $d_{\mathcal{D}_m}$ converges to the joint state distribution of π, d_π, with probability* 1*:*

$$\forall s,\ \lim_{m\to\infty} d_{\mathcal{D}_m}(s) = d_\pi(s).$$

Moreover, the empirical state visitation distribution for each agent i, $d_{\mathcal{D}_m,i}$, converges to the state visitation distribution of π_i, d_{π_i}, with probability 1*:*

$$\forall s_i,\ \lim_{m\to\infty} d_{\mathcal{D}_m,i}(s_i) = d_{\pi_i}(s_i) \quad \forall i \in \mathcal{I}.$$

In Theorem 2 below, the first result shows that CoSER decreases joint sampling error faster than on-policy sampling. We contrast this result with Example 2, which shows that MA-PROPS does not necessarily reduce joint sampling error and therefore lacks this guarantee. The second result of this theorem establishes a novel multi-agent result: this accelerated reduction in joint sampling error also guarantees that sampling error w.r.t. each agent decreases at the same accelerated rate. Because sub-optimal behavior by even a single agent (which can arise from sampling error) can cause all agents to converge to a sub-optimal joint policy (Zhao et al., 2023), reducing sampling error w.r.t. each agent *in addition to* reducing joint sampling error is important for reliable convergence.

Theorem 2. *Let s be a particular state that is visited m times during data collection and assume $|\mathcal{A}| \geq 2$. Then under Assumption 1,*

1. $D_{\mathrm{KL}}(\pi_{\mathcal{D}}(\cdot|s)\|\pi(\cdot|s)) = O_p\left(1/m^2\right)$ *under CoSER while* $D_{\mathrm{KL}}(\pi_{\mathcal{D}}(\cdot|s)\|\pi(\cdot|s)) = O_p\left(1/m\right)$ *under on-policy sampling.*
2. $D_{\mathrm{KL}}(\pi_{\mathcal{D},i}(\cdot|s)\|\pi_i(\cdot|s)) = O_p\left(1/m^2\right)$ *under CoSER while* $D_{\mathrm{KL}}(\pi_{\mathcal{D},i}(\cdot|s)\|\pi_i(\cdot|s)) = O_p\left(1/m\right)$ *under on-policy sampling.*

where O_p denotes stochastic boundedness.

6 Experiments

We design experiments to test the following hypotheses:

H1: CoSER achieves lower joint sampling error than MA-PROPS and on-policy sampling after collecting a fixed number of samples.

H2: CoSER yields more reliable on-policy policy gradient learning than MA-PROPS and on-policy sampling, increasing the fraction of training runs that converge optimally.

Evaluation metrics. To evaluate **H1**, we use two sampling error metrics. In tasks with small state-action dimensionality, we use the total variation (TV) distance between the empirical joint state-action visitation $d_{\mathcal{D}}(\boldsymbol{s}, \boldsymbol{a})$ distribution, denoting the proportion of times $(\boldsymbol{s}, \boldsymbol{a})$ appears in buffer $\mathcal{D}$, and the true joint state-action visitation distribution $d_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a})$ under $\pi_{\boldsymbol{\theta}}$: $d_{\text{TV}}(d_{\mathcal{D}}, d_{\pi_{\boldsymbol{\theta}}}) = \frac{1}{2}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} |d_{\mathcal{D}}(\boldsymbol{s}, \boldsymbol{a}) - d_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a})|$. In tasks where it is costly to compute $d_{\pi_{\boldsymbol{\theta}}}$, we follow Corrado & Hanna (2023) and Zhong et al. (2022) and compute sampling error as the KL divergence $D_{\text{KL}}(\pi_{\mathcal{D}}||\pi_{\boldsymbol{\theta}})$, where $\pi_{\mathcal{D}}(\boldsymbol{a}|\boldsymbol{s})$is the empirical policy denoting the fraction of times $\boldsymbol{a}$ was sampled at state $\boldsymbol{s}$ in $\mathcal{D}$. We estimate $\pi_{\mathcal{D}}$ as the maximum likelihood policy under $\mathcal{D}$ (see Appendix 10 for details). To evaluate **H2**, we track the *success rate*, the fraction of evaluation episodes in which the agents solve the task when sampling actions from the *decentralized* target policies. To ensure a fair comparison, we tune CoSER and MA-PROPS over the same hyperparameters and report results for the best setting (see Appendix 14 for details). In all figures, we plot the mean success rate or mean sampling error with 95% bootstrap confidence intervals.

Multi-agent games. We consider several multi-agent games from prior works: Level-Based Foraging (LBF) (Papoudakis et al., 2020a; Christianos et al., 2020), BoulderPush (Christianos et al., 2022), the 3×3 Climbing and Penalty matrix games, and all 21 structurally distinct 2×2 no-conflict matrix games (listed in Fig. 9 of Appendix 12) (Albrecht et al., 2024). We additionally create a custom 3x3 GridWorld (Fig. 3a). In all games, each agent has an optimal behavior, but the maximal return is achieved only when *all* agents act optimally; if some do so while others do not, all agents incur a penalty. In BoulderPush, agents must push a boulder to a goal, and both agents receive a penalty if only one agent attempts to push the boulder. Similarly, in LBF, agents must forage a food item together, and both agents receive a penalty if only one agent attempts to forage. In GridWorld, agents receive reward 0.9 if they both simultaneously reach the top-left cell (green) and reward -0.1 if only one does. Agents receive reward 0.1 if either agent reaches the bottom-right cell (blue). In all games except 2×2 matrix games 1–16, there exists at least one suboptimal equilibrium in which agents avoid acting optimally to avoid the risk of being penalized.[4] We provide additional game details in Appendix 12.

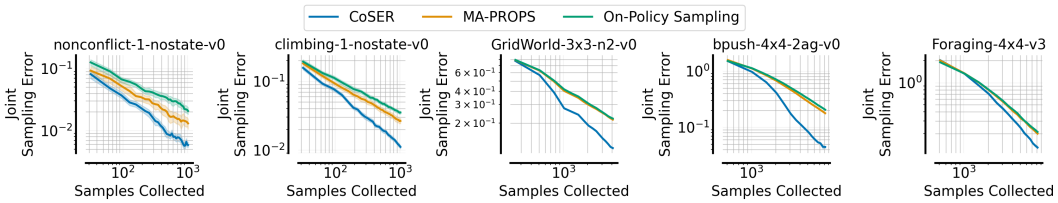
Reward rescaling. If the expected policy gradient points toward a sub-optimal equilibrium, agents will converge sub-optimally even if we have no joint sampling error. To isolate the effect of joint sampling error on convergence, we focus on games where *the expected policy gradient of each agent points toward optimality.* However, in most of these games, the expected policy gradient points away from optimality at initialization.[5] The probability of all agents cooperating is very small at initialization, so the expected return of any agent i acting optimally (*e.g.* pushing the boulder) has lower expected return than acting sub-optimally (*e.g.* avoiding the boulder) (Christianos et al., 2022; Zhao et al., 2023). Thus, we rescale the rewards in some games so that the expected policy gradient under uniformly initialized policies encourages cooperation. We detail these modifications in Appendix 12.

6.1 Reducing Sampling Error w.r.t. a Fixed Target Policy

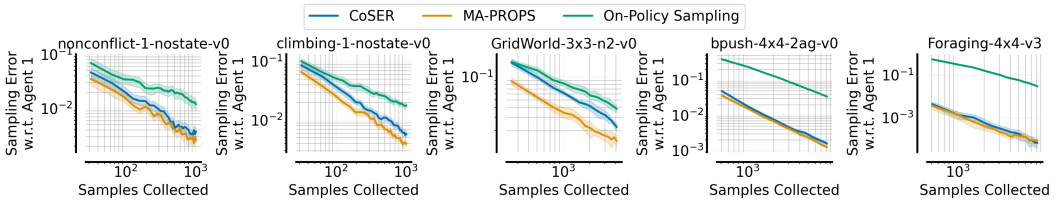
We first study how quickly CoSER decreases sampling error when each agent's policy is fixed. We provide two baselines: on-policy sampling and MA-PROPS. We randomly initialize each agent's policy and collect a fixed number of samples with each sampling method. For matrix games and GridWorld, we compute sampling error as $d_{\text{TV}}(d_{\mathcal{D}}, d_{\pi_{\boldsymbol{\theta}}})$. For all other tasks, we use $D_{\text{KL}}(\pi_{\mathcal{D}}||\pi_{\boldsymbol{\theta}})$.

[4] Albrecht et al. (2024) call this type of equilibrium *risk-dominant* because it has lower risk than the optimal equilibrium. Other works refer to it as *shadowed equilibrium*

[5] Christianos et al. (2022) show that on-policy policy gradient algorithms like MAPPO and MAA2C consistently converge sub-optimally in the same tasks we consider.



(a) Sampling error w.r.t. the joint policy.



(b) Sampling error w.r.t. Agent 1's policy.

Figure 4: Mean sampling error over 10 seeds with 95% bootstrap confidence intervals. Takeaway: CoSER reduces joint sampling error with fewer samples than MA-PROPS and on-policy sampling even though MA-PROPS often reduces sampling error w.r.t. Agent 1 faster than CoSER.

Since all matrix games have the same dynamics, we focus on 2×2 game 1 and the 3×3 Climbing game. As shown in Fig. 4a, CoSER consistently achieves lower joint sampling error than both MA-PROPS and on-policy sampling. MA-PROPS yields only marginal improvements over on-policy sampling in most tasks. Fig. 4b shows that CoSER achieves the lowest joint sampling error even though MA-PROPS decreases sampling error w.r.t. each agent more than CoSER, highlighting how reducing sampling error w.r.t. each agent does not guarantee reduced joint sampling error. These results support **H1**.

6.2 Reducing Sampling Error During Reinforcement Learning

We now examine how CoSER affects the reliability of on-policy policy gradient algorithms. We train agents using CoSER, MA-PROPS, and on-policy sampling and track their success rate and joint sampling error over training. Our main experiments use MAPPO (Yu et al., 2022) to update target policies. We report additional results using IPPO (De Witt et al., 2020) in Appendix 13. We compute sampling error as $d_{\text{TV}}(d_{\mathcal{D}}, d_{\pi_{\boldsymbol{\theta}}})$ for matrix games and $D_{\text{KL}}(\pi_{\mathcal{D}} || \pi_{\boldsymbol{\theta}})$ for all other tasks. Since CoSER and MA-PROPS generate different target policy sequences during training, we compute on-policy sampling error separately for each method by filling a second buffer with samples from the corresponding target policies.

GridWorld, BoulderPush, LBF. As shown in Fig. 5, CoSER has a higher probability of converging optimally in GridWorld, BoulderPush, and LBF compared to on-policy sampling and MA-PROPS, supporting **H2**. At convergence, CoSER improves success rate over on-policy sampling by 15 percentage points in GridWorld, 19 in BoulderPush, and 10 in LBF. In contrast, MA-PROPS provides only marginal benefit in LBF, no improvement in BoulderPush, and a 20-point drop in GridWorld. Fig. 6d, 6e, and 6f show joint sampling error during training for GridWorld, BoulderPush, and LBF and clarifies why CoSER outperforms MA-PROPS: CoSER decreases joint sampling error more than MA-PROPS. In fact, MA-PROPS does not improve over on-policy sampling at all in LBF. These results support **H1**.

Matrix Games. We show training curves for 2×2 games 19-21 and both 3×3 games in Fig. 5. In these games, CoSER is the only algorithm that consistently converges to the optimal policy. We provide training curves for the remaining 2×2 games in Appendix 13. In games 1–18, MA-PROPS and on-policy sampling achieve success rates of at least 95% of runs. This result is consistent

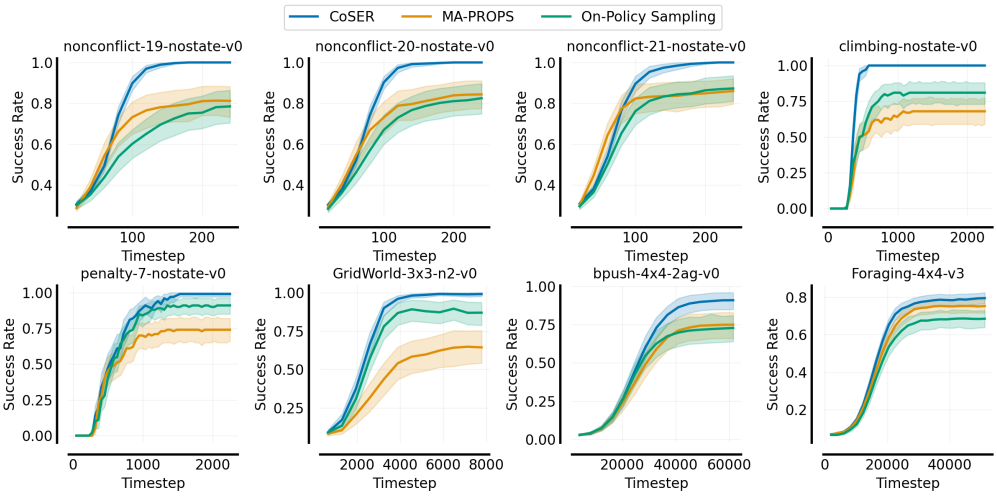


Figure 5: MAPPO mean success rates over 100 seeds with 95% bootstrap confidence intervals. Takeaway: CoSER increases success rate by 10-20% over on-policy sampling and MA-PROPS. MA-PROPS only increases success rate in LBF.

with findings by Christianos et al. (2022), who observed that independent on-policy policy gradient algorithms like MAA2C only struggle in games 19–21. Nevertheless, CoSER improves reliability even in these easier games, achieving perfect success rates in all games and also converging faster in games 6, 11, 12, 16, and 17. These results support **H2**.

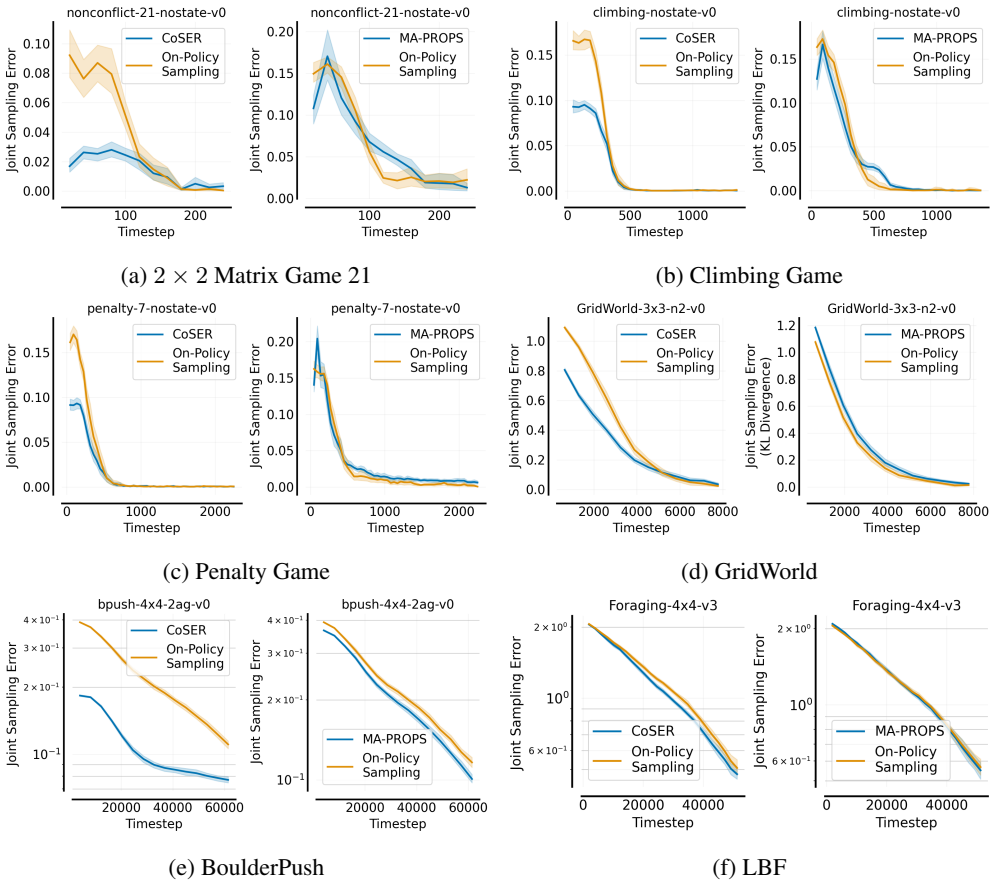
Fig. 6a, 6b, and 6c show joint sampling error curves for 2×2 matrix game 21 and the 3×3 Climbing and Penalty games. Due to space constraints, we provide joint sampling error curves for the remaining 2×2 matrix games in Fig. 12 of Appendix 13. In all matrix games, results are qualitatively similar: CoSER decreases joint sampling error by a large margin before learning converges, while MA-PROPS generally does not decrease joint sampling error. Both CoSER and MA-PROPS reduce sampling error w.r.t. Agent 1, again highlighting how reducing sampling error w.r.t. each agent may not reduce joint sampling error. These results support **H1**.

7 Limitations

The core goal of this paper is to characterize a subtle failure mode of independent on-policy policy gradient algorithms—joint sampling error—and then demonstrate that reducing joint sampling error via adaptive action sampling can improve the reliability of these algorithms. In this section, we discuss limitations of our proposed approach (CoSER) and outline directions for future work.

Scaling to many agents. Since the joint action space grows exponentially with the number of agents, it may be difficult to learn a centralized behavior policy for tasks with many agents. A potential solution is to use deep coordination graphs (DCG) (Böhmer et al., 2020) to factor the behavior policy into locally conditioned components, trading off representational capacity for scalability.

Batch size. CoSER increases the probability of under-sampled joint actions at states $\boldsymbol{s}$ in $\mathcal{D}$ so that sampling error is reduced *upon the agents' next visit to states with features similar to the features of* $\boldsymbol{s}$. If $\mathcal{D}$ is small, then revisits to states with features similar to $\boldsymbol{s}$ are rare, and CoSER will perform similarly to on-policy sampling. This could be addressed by letting $\mathcal{D}$ retain historic data between target policy updates. Corrado & Hanna (2023) showed that PROPS can leverage historic data by collecting additional data so that the aggregate distribution in $\mathcal{D}$ is close to on-policy. Since CoSER shares the same behavior update as PROPS, it can likely benefit from the same data reuse strategies shown to be effective in single-agent settings.



(a) 2×2 Matrix Game 21

(b) Climbing Game

(c) Penalty Game

(d) GridWorld

(e) BoulderPush

(f) LBF

Figure 6: Mean joint sampling error over 100 seeds with 95% bootstrap confidence intervals. Takeaway: CoSER reduces joint sampling error more than MA-PROPS.

Discrete Actions. Our experiments focus on discrete action tasks, so the centralized behavior policy we describe in Section 5.2 assumes discrete actions. However, this architecture extends in principle to continuous action settings. We detail this extension in Appendix 11.

8 Conclusion

In this paper, we first identify a subtle failure mode of independent on-policy policy gradient algorithms. Stochasticity in action sampling can cause the joint distribution of collected data to deviate from the expected joint on-policy distribution, and this *sampling error* can cause agents to converge to sub-optimal solutions—*even when the expected policy gradients align with optimal behavior*. Given this failure mode, we asked: Can reducing sampling error in the joint data distribution via coordinated action selection lead to more reliable convergence of on-policy policy gradient algorithms? Toward answering this question, we introduce **Co**operative **S**ampling **E**rror **R**eduction (CoSER), an action sampling algorithm that adaptively corrects joint sampling error during multi-agent on-policy data collection. CoSER follows the Centralized Training with Decentralized Execution (CTDE) paradigm: agents collect data using a centralized behavior policy that we continually adapt to increase the probability of under-sampled joint actions, and this data is then used to train decentralized policies for deployment. Empirically, CoSER reduces sampling error in the joint on-policy distribution more sample-efficiently than standard on-policy sampling, and increases the fraction of training runs that converge optimally.

Acknowledgments

We thank Brahma Pavse and Adam Labiosa for valuable feedback on earlier drafts. This work took place in the Prediction and Action Lab (PAL) at the University of Wisconsin – Madison. PAL research is supported by the National Science Foundation (IIS-2410981) and the Wisconsin Alumni Research Foundation.

References

Stefano V Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2020.

Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10707–10717, 2020.

Filippos Christianos, Georgios Papoudakis, and Stefano V Albrecht. Pareto actor-critic for equilibrium selection in multi-agent reinforcement learning. *arXiv preprint arXiv:2209.14344*, 2022.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

Nicholas E Corrado and Josiah P Hanna. On-policy policy gradient reinforcement learning without on-policy sampling. *arXiv preprint arXiv:2311.08290*, 2023.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Nancy Fulda and Dan Ventura. Predicting and preventing coordination problems in cooperative q-learning systems. In *IJCAI*, volume 2007, pp. 780–785, 2007.

Tarun Gupta, Anuj Mahajan, Bei Peng, Wendelin Böhmer, and Shimon Whiteson. Uneven: Universal value exploration for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 3930–3941. PMLR, 2021.

Josiah P Hanna, Scott Niekum, and Peter Stone. Importance sampling in reinforcement learning with an estimated behavior policy. *Machine Learning*, 110(6):1267–1317, 2021.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Ksenia Konyushova, Yutian Chen, Thomas Paine, Caglar Gulcehre, Cosmin Paduraru, Daniel J Mankowitz, Misha Denil, and Nando de Freitas. Active offline policy selection. *Advances in Neural Information Processing Systems*, 34:24631–24644, 2021.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2021.

Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. In *Artificial Intelligence and Statistics*, pp. 608–616. PMLR, 2015.

Pengyi Li, Hongyao Tang, Tianpei Yang, Xiaotian Hao, Tong Sang, Yan Zheng, Jianye Hao, Matthew E Taylor, Wenyuan Tao, Zhen Wang, et al. Pmic: Improving multi-agent reinforcement learning with progressive mutual information collaboration. *arXiv preprint arXiv:2203.08553*, 2022.

Michael L Littman et al. Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, pp. 322–328, 2001.

Iou-Jen Liu, Unnat Jain, Raymond A Yeh, and Alexander Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In *International conference on machine learning*, pp. 6826–6836. PMLR, 2021.

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402*, 2021.

Yanhao Ma and Jie Luo. Value-decomposition multi-agent proximal policy optimization. In *2022 China Automation Congress (CAC)*, pp. 3460–3464. IEEE, 2022.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in neural information processing systems*, 32, 2019.

Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 64–69. IEEE, 2007.

Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. Policy optimization via importance sampling. *Advances in Neural Information Processing Systems*, 31, 2018.

Subhojyoti Mukherjee, Josiah P Hanna, and Robert D Nowak. Revar: Strengthening policy evaluation via reduced variance sampling. In *Uncertainty in Artificial Intelligence*, pp. 1413–1422. PMLR, 2022.

Yusuke Narita, Shota Yasui, and Kohei Yata. Efficient counterfactual learning from bandit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4634–4641, 2019.

Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. *arXiv preprint arXiv:1707.04402*, 2017.

Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In *International conference on machine learning*, pp. 4026–4035. PMLR, 2018.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020a.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Comparative evaluation of cooperative multi-agent deep reinforcement learning algorithms. *arXiv preprint arXiv:2006.07869*, 2020b.

Brahma S. Pavse, Ishan Durugkar, Josiah P. Hanna, and Peter Stone. Reducing sampling error in batch temporal difference learning. In *International Conference on Machine Learning*, 2020.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pp. 5887–5896. PMLR, 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Aaron David Tucker and Thorsten Joachims. Variance-optimal augmentation logging for counterfactual evaluation in contextual bandits. *arXiv preprint arXiv:2202.01721*, 2022.

Runzhe Wan, Branislav Kveton, and Rui Song. Safe exploration for efficient policy evaluation and comparison. In *International Conference on Machine Learning*, pp. 22491–22511. PMLR, 2022.

Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. Influence-based multi-agent exploration. *arXiv preprint arXiv:1910.05512*, 2019.

Ermo Wei and Sean Luke. Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research*, 17(84):1–42, 2016.

Rudolf Paul Wiegand. *An analysis of cooperative coevolutionary algorithms*. George Mason University, 2004.

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.

Shaowei Zhang, Jiahan Cao, Lei Yuan, Yang Yu, and De-Chuan Zhan. Self-motivated multi-agent exploration. *arXiv preprint arXiv:2301.02083*, 2023.

Wenshuai Zhao, Yi Zhao, Zhiyuan Li, Juho Kannala, and Joni Pajarinen. Optimistic multi-agent policy gradient. *arXiv preprint arXiv:2311.01953*, 2023.

Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *Advances in Neural Information Processing Systems*, 34:3757–3769, 2021.

Rujie Zhong, Duohan Zhang, Lukas Schäfer, Stefano Albrecht, and Josiah Hanna. Robust on-policy sampling for data-efficient policy evaluation in reinforcement learning. *Advances in Neural Information Processing Systems*, 35:37376–37388, 2022.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024.

Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. Smarts: An open-source scalable multi-agent rl training school for autonomous driving. In *Conference on robot learning*, pp. 264–285. PMLR, 2021.

Supplementary Materials

The following content was not necessarily subject to peer review.

9 Convergence Analysis

In this section, we present the proof of Theorems 1 and 2 from the main paper. The proofs rest on Assumption 2 and Theorem 2 by Zhong et al. (2022) as well as Proposition 1 by Corrado & Hanna (2023), which we first restate below for completeness. Recall that ROS is the algorithm introduced by Zhong et al. (2022) to reduce sampling error w.r.t. a single policy in the single-agent setting.

Assumption 2 (Assumption 2 in Zhong et al. (2022))**.** *ROS uses a step-size of $\alpha \to \infty$ and the behavior policy is parameterized as a softmax function, i.e., $\pi_\phi(a|s) \propto e^{\phi_{s,a}}$, where for each state $\boldsymbol{s}$ and action $\boldsymbol{a}$, we have a parameter $\boldsymbol{\phi}_{s,a}$. This assumption implies that ROS always selects the most under-sampled joint action in each state.*

Theorem 3 (Theorem 1 in Zhong et al. (2022))**.** *Let s be a particular state that is visited m times during data collection and assume that $|\mathcal{A}| \geq 2$. Under Assumption 2, $D_{\mathrm{KL}}(\pi_{\mathcal{D}}(\cdot|s)\|\pi(\cdot|s)) = O_p\left(\frac{1}{m^2}\right)$ under ROS sampling while $D_{\mathrm{KL}}(\pi_{\mathcal{D}}(\cdot|s)\|\pi(\cdot|s)) = O_p\left(\frac{1}{m}\right)$ under on-policy sampling where O_p denotes stochastic boundedness.*

Theorem 4 (Proposition 1 in Corrado & Hanna (2023))**.** *Let $\boldsymbol{s}$ be a state that we visit m times. Under Assumption 2, we have $\forall a \in \mathcal{A}$ that:*

$$\lim_{m\to\infty} \pi_{\mathcal{D}}(a|s) = \pi(a|s).$$

We now prove similar results for CoSER. We first make an assumption similar to Assumption 2 posed by Zhong et al. (2022).

Assumption 3 (Restated Assumption 1)**.** *CoSER uses a learning rate of $\alpha \to \infty$ and the behavior policy is parameterized as a softmax function, i.e., $\pi_\phi(a|s) \propto e^{\phi_{s,a}}$, where for each state $\boldsymbol{s}$ and action $\boldsymbol{a}$, we have a parameter $\boldsymbol{\phi}_{s,a}$. This assumption implies that CoSER always selects the most under-sampled joint action in each state.*

To see why this assumption implies that CoSER always selects the most under-sampled joint action in each state, recall that at the start of every behavior policy update, we have $\pi_{\boldsymbol{\phi}} \equiv \pi_{\boldsymbol{\theta}}$ so that $\pi_{\boldsymbol{\phi}}(a|s)/\pi_{\boldsymbol{\theta}}(a|s) = 1$ for all (s, a). Thus, clipping is not applied, and the CoSER gradient reduces to the ROS gradient:

$$\begin{aligned}
\frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} \nabla_{\boldsymbol{\phi}}\mathcal{L}(\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{s},\boldsymbol{a},\varepsilon) &= \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} \nabla_{\boldsymbol{\phi}}\left(-\frac{\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})}\right) \\
&= \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} -\nabla_{\boldsymbol{\phi}}\log\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})\frac{\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \\
&= \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} -\nabla_{\boldsymbol{\phi}}\log\pi_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{s})
\end{aligned} \tag{6}$$

Thus, since Assumption 2 implies that ROS samples the most under-sampled action, so does CoSER.

Our first theorem shows how empirical state visitation distributions converge to their true state visitation distributions under CoSER. We use $d_{\mathcal{D}_m}$ and $\pi_{\mathcal{D}_m}$, as the empirical state visitation distribution and empirical policy after m state-action pairs have been taken, respectively. That is, $d_{\mathcal{D}_m}(s)$ is the proportion of the m states that are s, $\pi_{\mathcal{D}_m}(a|s)$ is the proportion of the time that action a was observed in state s. We use subscript i to denote analogous quantities for agent i.

Theorem 5 (Restated Theorem 1)**.** *Assume that $\mathcal{S}$ and $\mathcal{A}$ are finite. Under CoSER with Assumption 3, the empirical joint state visitation distribution, $d_{\mathcal{D}_m}$, converges to the joint state distribution of π, d_π, with probability* 1*:*

$$\forall s, \lim_{m\to\infty} d_{\mathcal{D}_m}(s) = d_\pi(s).$$

Moreover, the empirical state visitation distribution for each agent i, $d_{\mathcal{D}_m,i}$, converges to the state distribution of π_i, d_{π_i}, with probability 1*:*

$$\forall s_i, \lim_{m\to\infty} d_{\mathcal{D}_m,i}(s_i) = d_{\pi_i}(s_i) \quad \forall i \in \mathcal{I}.$$

Proof. Since the behavior policy $\pi(a|s) = \prod_{i=1}^n \pi_i(a|s_i)$ is a single agent mapping joint states to joint actions, we can immediately apply Theorem 4 to obtain the result for the joint state visitation distribution. The result for each agent's state visitation distribution follows from marginalizing the joint state visitation distribution:

$$\lim_{m\to\infty} d_{\mathcal{D}_m}(s_i) = \lim_{m\to\infty} \sum_{s_{-i}} d_{\mathcal{D}_m}(s) = \sum_{s_{-i}} \lim_{m\to\infty} d_{\mathcal{D}_m}(s) = \sum_{s_{-i}} d_\pi(s) = d_{\pi_i}(s_i)$$

□

Our second theorem shows that joint sampling error decreases faster under CoSER than under on-policy sampling. We further establish a novel multi-agent result: this accelerated reduction in joint sampling error also guarantees that sampling error w.r.t. each agent decreases at the same accelerated rate. We let π_i denote the policy of agent i and let $\pi_{\mathcal{D},i}$ denote the empirical policy of agent i.

Theorem 6 (Restated Theorem 2)**.** *Let s be a particular state that is visited m times during data collection and assume that $|\mathcal{A}| \geq 2$. Under Assumption 1, we have*

1. *$D_{\mathrm{KL}}(\pi_{\mathcal{D}}(\cdot|s)\|\pi(\cdot|s)) = O_p\left(\frac{1}{m^2}\right)$ under CoSER while $D_{\mathrm{KL}}(\pi_{\mathcal{D}}(\cdot|s)\|\pi(\cdot|s)) = O_p\left(\frac{1}{m}\right)$ under on-policy sampling*
2. *$D_{\mathrm{KL}}(\pi_{\mathcal{D},i}(\cdot|s)\|\pi_i(\cdot|s)) = O_p\left(\frac{1}{m^2}\right)$ under CoSER while $D_{\mathrm{KL}}(\pi_{\mathcal{D},i}(\cdot|s)\|\pi_i(\cdot|s)) = O_p\left(\frac{1}{m}\right)$ under on-policy sampling*

where O_p denotes stochastic boundedness.

Proof. Since the behavior policy $\pi(a|s) = \prod_{i=1}^n \pi_i(a|s_i)$ is a single agent mapping joint states to joint actions, we can immediately apply the convergence result from Theorem 3 to obtain the convergence result for joint sampling error follows. Since $D_{\mathrm{KL}}(\pi_{\mathcal{D}}(\cdot|s)\|\pi(\cdot|s)) \geq D_{\mathrm{KL}}(\pi_{\mathcal{D},i}(\cdot|s)\|\pi_i(\cdot|s))$ for all agents $i \in \mathcal{I}$, the result follows from the result for joint sampling error. □

Remark 1. *This result implies that no individual agent is disproportionately affected by sampling error under CoSER. Because sub-optimal behavior by even a single agent (which can arise from sampling error) can cause all agents to converge to a sub-optimal joint policy (Zhao et al., 2023), reducing sampling error w.r.t. each agent is important for reliable convergence.*

10 Computing Sampling Error

Similar to Zhong et al. (2022) and Corrado & Hanna (2023), we measure sampling error as the KL divergence $D_{\text{KL}}(\pi_{\mathcal{D}}||\pi_{\boldsymbol{\theta}})$ between the empirical joint policy $\pi_{\mathcal{D}}$ and the target joint policy $\pi_{\boldsymbol{\theta}}$:

$$D_{\text{KL}}(\pi_{\mathcal{D}}||\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{\boldsymbol{s}\sim\mathcal{D},\boldsymbol{a}\sim\pi_{\mathcal{D}}(\cdot|\boldsymbol{s})}\left[\log\left(\frac{\pi_{\mathcal{D}}(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})}\right)\right]. \tag{7}$$

We compute a parametric estimate of $\pi_{\mathcal{D}}$ by letting $\boldsymbol{\theta}'$ be the parameters of a neural network that takes joint states as input and outputs a distribution over joint actions $\pi_{\boldsymbol{\theta}'}(\cdot|\boldsymbol{s})$ and maximizing the log-likelihood of $\mathcal{D}$ under $\pi'_{\boldsymbol{\theta}}$

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}'} \sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} \log \pi_{\boldsymbol{\theta}'}(\boldsymbol{a}|\boldsymbol{s}) \tag{8}$$

using stochastic gradient ascent. After computing $\boldsymbol{\theta}_{\text{MLE}}$, we then estimate sampling error using the Monte Carlo estimator:

$$D_{\text{KL}}(\pi_{\mathcal{D}}||\pi_{\boldsymbol{\theta}}) \approx \sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} \left(\log \pi_{\boldsymbol{\theta}_{\text{MLE}}}(\boldsymbol{a}|\boldsymbol{s}) - \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})\right). \tag{9}$$

We compute sampling error w.r.t. individual agents in a similar fashion: We compute estimate $\pi_{\mathcal{D}_i}$ as the maximum likelihood policy under $\mathcal{D}_i$ and then estimate sampling error using the Monte Carlo estimator:

$$D_{\text{KL}}(\pi_{\mathcal{D}_i}||\pi_{\boldsymbol{\theta}_i}) \approx \sum_{(\boldsymbol{s}_i,\boldsymbol{a}_i)\in\mathcal{D}} \left(\log \pi_{\boldsymbol{\theta}_{i,\text{MLE}}}(\boldsymbol{a}_i|\boldsymbol{s}_i) - \log \pi_{\boldsymbol{\theta}_i}(\boldsymbol{a}_i|\boldsymbol{s}_i)\right). \tag{10}$$

11 Extending CoSER to Continuous Action Tasks

The centralized behavior policy we use is specific to softmax policies for discrete action settings. However, this architecture easily extends to continuous action settings. Continuous policies are typically parameterized as Gaussians $\mathcal{N}(\boldsymbol{\mu}_i(\boldsymbol{s}_i), \sigma_i(\boldsymbol{s}_i))$ with mean $\boldsymbol{\mu}_i(\boldsymbol{s}_i)$ and variance $\sigma_i(\boldsymbol{s}_i)$. The joint policy is then a Gaussian with mean $\boldsymbol{\mu}(\boldsymbol{s}) = (\boldsymbol{\mu}_1(\boldsymbol{s}_1), \ldots, \boldsymbol{\mu}_n(\boldsymbol{s}_n))$ and variance $\sigma(\boldsymbol{s}) = (\sigma_1(\boldsymbol{s}_1), \ldots, \sigma_n(\boldsymbol{s}_n))$. Similar to how the behavior network adds an adjustment to the joint logits in the discrete setting, in a continuous setting, the behavior network would output an adjustment to the joint mean and joint variance.

More concretely, we first compute the mean $\boldsymbol{\mu}(\boldsymbol{s})$ and variance $\sigma(\boldsymbol{s})$ of the joint policy. Next, we define a neural network $\Delta_{\boldsymbol{\phi}}(\boldsymbol{s}) : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$ that outputs an adjustment to the mean and variance of each joint action dimension. Let $\Delta^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{s})$ denote the mean adjustments and let $\Delta^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{s})$ denote the variance adjustments. The behavior policy is then[6]

$$\pi_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s}) = \mathcal{N}\left(\mu(\boldsymbol{s}) + \Delta^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{s}), \sigma(\boldsymbol{s}) + \Delta^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{s})\right) \tag{11}$$

To initialize behavior policy to match $\pi_{\boldsymbol{\theta}}$ at the start of each update, we set the final layer of $\Delta_{\boldsymbol{\phi}}$ to the zero vector so that $\Delta^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{s}) = \mathbf{0}$ and $\Delta^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{s}) = \mathbf{0}$ for all $\boldsymbol{s} \in \mathcal{S}$. Then, we perform minibatch gradient ascent on $\mathcal{L}(\boldsymbol{\phi})$ to adapt the logit adjustment so that the behavior policy places larger probability on under-sampled actions just as described in Algorithm 2.

[6] We again omit the behavior policy's dependence on $\boldsymbol{\theta}$ for clarity, as $\boldsymbol{\theta}$ is fixed during behavior updates.

		Agent 2		
		A	B	C
Agent 1	A	$11^\star$	-30	0
	B	-30	$7^\dagger$	0
	C	0	6	5

(a) Original Climbing game (Claus & Boutilier, 1998).

		Agent 2		
		A	B	C
Agent 1	A	$11^\star$	-3	0
	B	-3	$7^\dagger$	0
	C	0	3	2

(b) Modified Climbing game.

		Agent 2		
		A	B	C
Agent 1	A	$-k$	0	$10^\star$
	B	0	$2^\dagger$	0
	C	$10^\star$	0	$-k$

(a) Original Penalty game (Claus & Boutilier, 1998).

		Agent 2		
		A	B	C
Agent 1	A	-7	0	$10^\star$
	B	0	$2^\dagger$	0
	C	$10^\star$	0	-7

(b) Modified Penalty game. We choose $k = 7$.

Figure 8: Original and modified 3×3 matrix games, where $\star$ denotes optimal equilibria, and $\dagger$ denotes suboptimal equilibria.

12 Multi-Agent Games

In this appendix, we further describe each multi-agent game we use. We additionally detail how we modify some games to ensure the expected policy gradient encourages cooperation at initialization.

12.1 Game Descriptions

BoulderPush: In this game, agents must coordinate to push a boulder to a specified goal position. Agents have four actions that move them one position up, down, left, or right. To push the boulder, all agents must move to the positions above the boulder (*i.e.*, the cells indicating the direction in which the boulder must be pushed) and then move in the indicated direction. Agents receive reward 0.1 for successfully pushing the boulder. If only one agent attempts to push the boulder independently (without the other agent), they receive reward -0.02.

Level-based Foraging: In this game, agents must coordinate to collect food items on the game board. Agents have five actions; four actions can move them one position up, down, left, or right. The fifth action is the "forage" action where the agent attempts to forage a food item. To forage the food, both agents must move next to it and choose the forage action. Agents receive reward 0.5 for successfully foraging the food. If only one agent attempts to forage independently (without the other agent), they receive reward -0.015.

12.2 Reward Modifications

Our work focuses on improving the reliability of on-policy policy gradient algorithms *when the expected policy gradient of each agent points toward optimality at initialization.* However, in most of these games, the expected policy gradient points away from optimality at initialization.[7] The probability of all agents cooperating is very small at initialization, so the expected return of any agent i acting optimally (*e.g.* pushing the boulder) has lower expected return than acting sub-optimally

[7] Christianos et al. (2022) show that on-policy policy gradient algorithms like MAPPO and MAA2C consistently converge sub-optimally in the same tasks we consider.

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$3, 3$
B	$2, 2$	$1, 1$

Game 1

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$3, 3$
B	$2, 1$	$2, 1$

Game 2

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$3, 2$
B	$2, 3$	$1, 1$

Game 3

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$3, 2$
B	$3, 2$	$1, 1$

Game 4

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$3, 1$
B	$2, 1$	$1, 3$

Game 5

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$3, 3$
B	$2, 1$	$1, 2$

Game 6

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 5}$	$3, 2$
B	$1, 1$	$1, 1$

Game 7

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 5}$	$3, 2$
B	$1, 1$	$2, 3$

Game 8

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 5}$	$3, 2$
B	$2, 3$	$1, 1$

Game 9

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 5}$	$3, 1$
B	$1, 1$	$2, 2$

Game 10

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 5}$	$3, 1$
B	$1, 1$	$2, 3$

Game 11

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 5}$	$3, 1$
B	$2, 3$	$2, 1$

Game 12

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$2, 3$
B	$3, 1$	$1, 3$

Game 13

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$2, 3$
B	$3, 1$	$2, 2$

Game 14

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$2, 2$
B	$3, 1$	$1, 3$

Game 15

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$2, 2$
B	$3, 2$	$\underline{1, 3}$

Game 16

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$3, 1$
B	$2, 2$	$\underline{1, 3}$

Game 17

Agent 1 \ Agent 2	A	B
A	$\mathbf{4, 4}$	$2, 1$
B	$1, 2$	$\underline{3, 3}$

Game 18

Agent 1 \ Agent 2	A	B
A	$\mathbf{5, 5}$	$1, 3$
B	$3, 1$	$\underline{2, 2}$

Game 19

Agent 1 \ Agent 2	A	B
A	$\mathbf{5, 5}$	$1, 2$
B	$3, 1$	$\underline{2, 2}$

Game 20

Agent 1 \ Agent 2	A	B
A	$\mathbf{5, 5}$	$1, 2$
B	$2, 1$	$\underline{3, 3}$

Game 21

Figure 9: All structurally distinct 2×2 no-conflict matrix games from Section 11.2.1 of Albrecht et al. (2024). Each cell shows the reward pair (r_1, r_2) for Agents 1 and 2. We bold the welfare-optimal Nash equilibrium and underline any welfare-suboptimal Nash equilibria. In all games, the optimal outcome is (A, A). To ensure the true policy gradient w.r.t. uniformly random policies increases the probability of the optimal outcome, we change the reward associated with the optimal outcome to $(4, 5)$ in games 7-12 and $(5, 5)$ in games 19-21.

(*e.g.* avoiding the boulder) (Christianos et al., 2022; Zhao et al., 2023). To ensure the task setting aligns with our problem of interest, we rescale the reward function in some environments so that the expected policy gradient under uniformly initialized policies encourages cooperation. Without this adjustment, these games would not reflect the failure mode we aim to study and would be unsuitable for testing our hypotheses.

- **LBF:** We reduce the penalty for failed cooperation from -0.1 to -0.015.
- **BoulderPush:** We reduce the penalty for failed cooperation from -0.1 to -0.02.
- 3×3 **Climbing game:** We show the original Climbing game rewards in Fig. 7a and our modified Climbing game in Fig. 7b.
- 3×3 **Penalty game:** We show the original Penalty game rewards in Fig. 8a and our modified Penalty game in Fig. 8b.
- 2×2 **no-conflict matrix games:** In all games, the optimal outcome originally has reward $4, 4$. In games 7-12, we change this reward to $4, 5$. In games 19-21, we change it to $5, 5$.

13 Additional Experiments

In this section, we provide additional experiments excluded from the main paper due to space constraints:

1. Training curves for all 21 distinct 2×2 no-conflict matrix games using MAPPO (Fig. 10) and IPPO (Fig. 11)
2. Sampling error curves for RL training in all 21 distinct 2×2 no-conflict matrix games (Fig. 12 and Fig. 13).
3. Training curves for BoulderPush and Level-based foraging tasks using IPPO (Fig. 14).

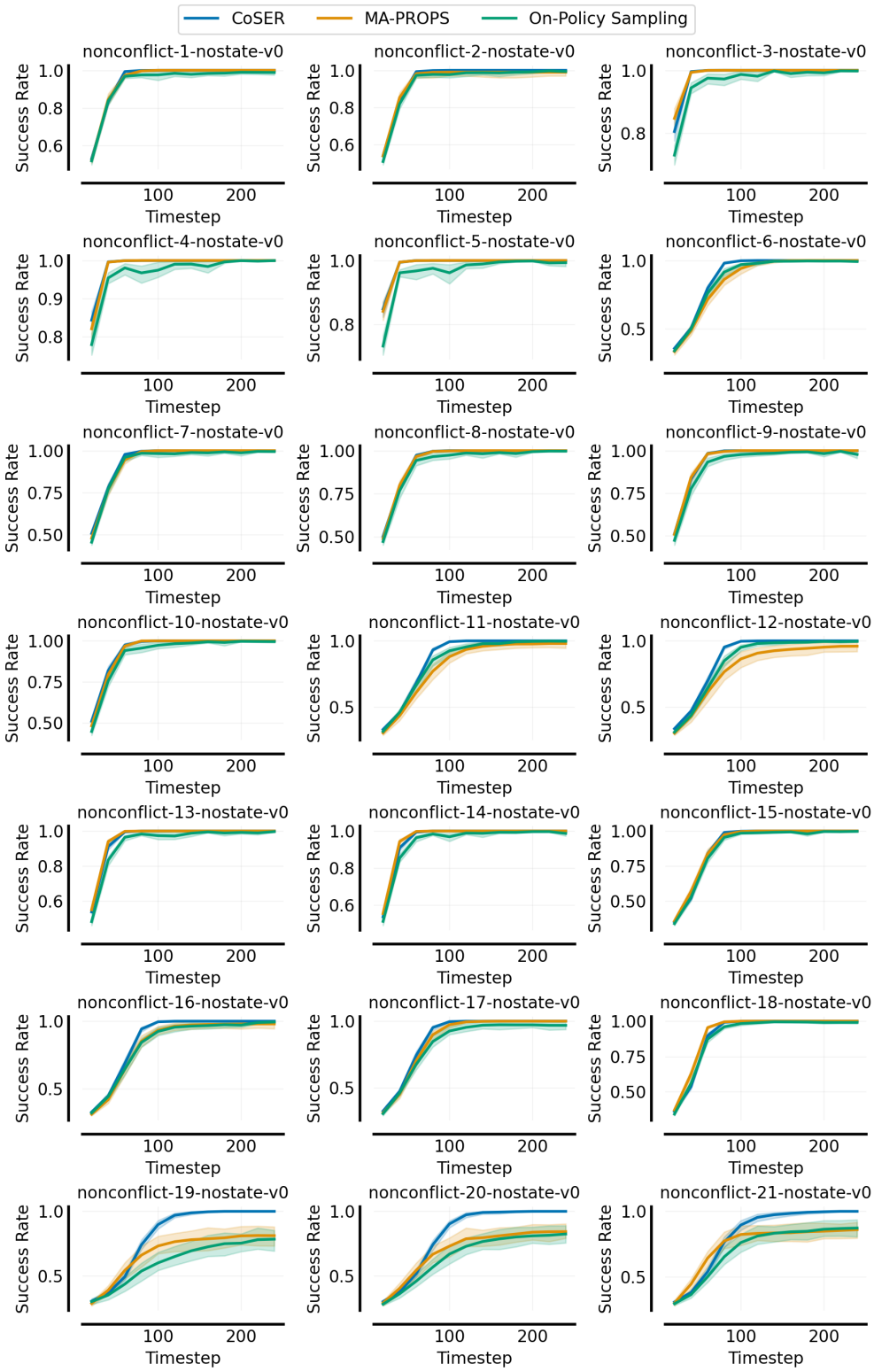


Figure 10: MAPPO mean success rate during training for all structurally distinct 2×2 no-conflict matrix games. Solid curves denote means over 100 seeds. Shaded regions denote 95% bootstrap confidence intervals.

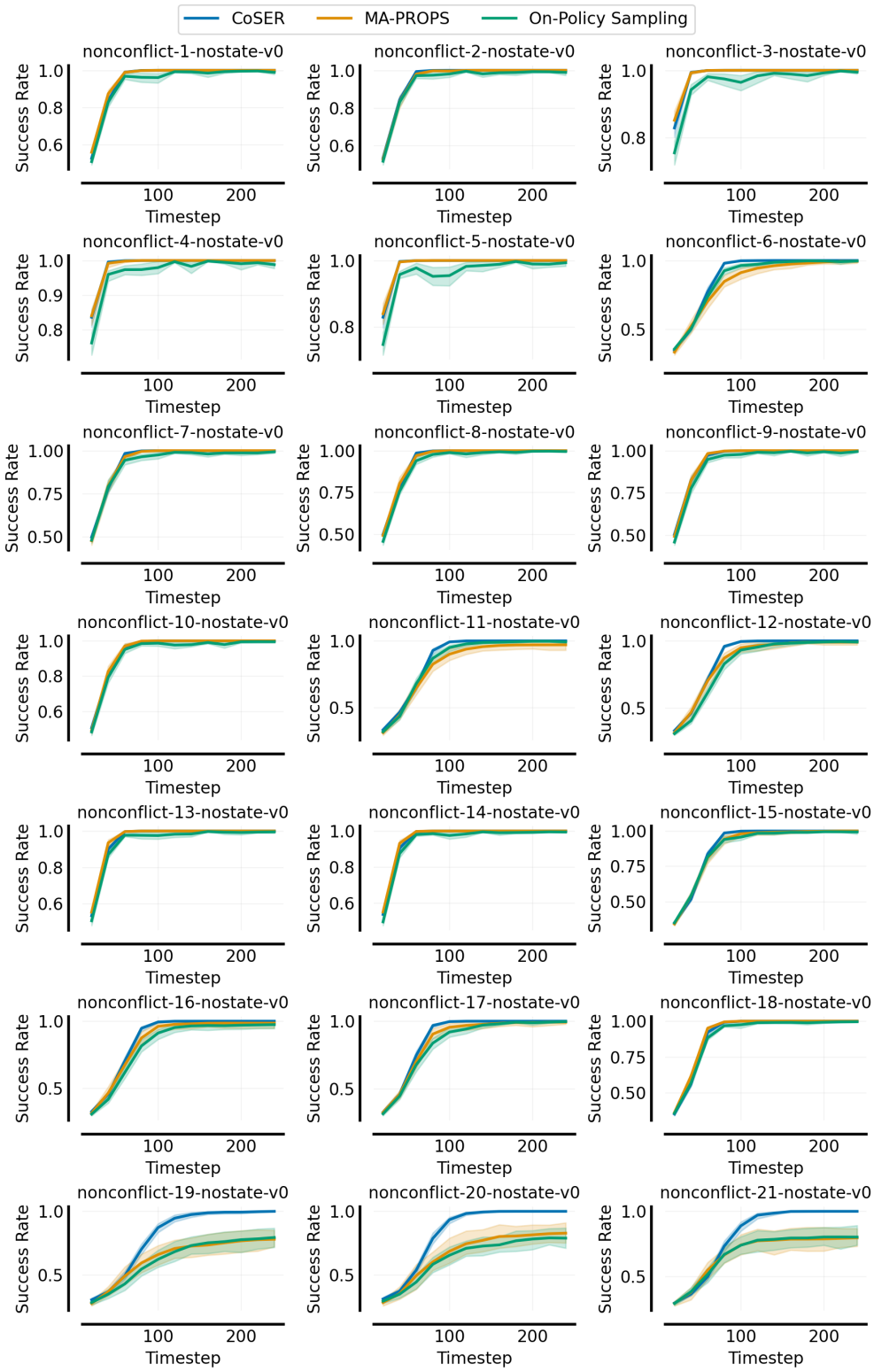


Figure 11: IPPO mean success rate during training for all structurally distinct 2×2 no-conflict matrix games. Solid curves denote means over 100 seeds. Shaded regions denote 95% bootstrap confidence intervals.

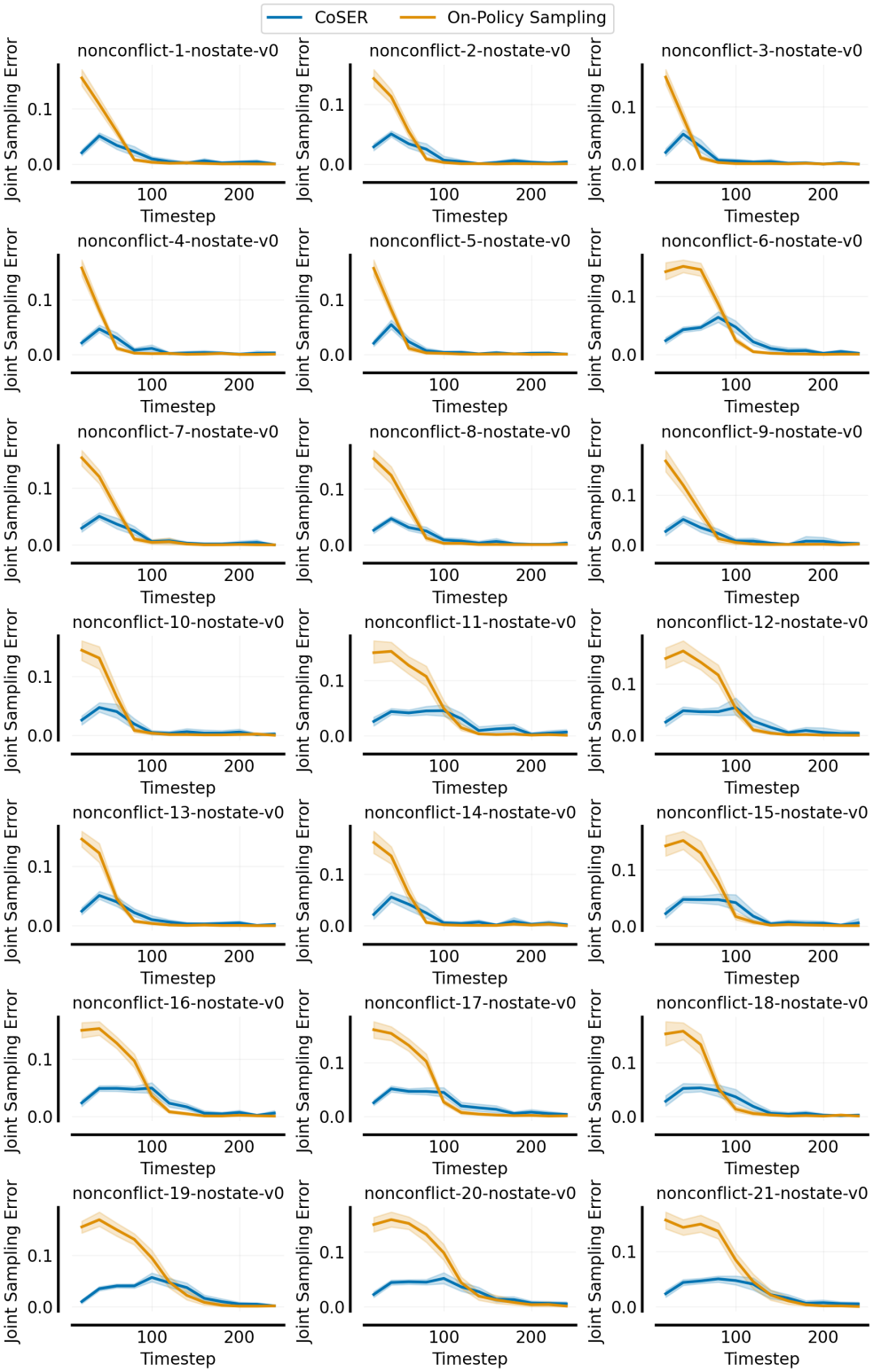


Figure 12: Mean joint sampling error for MAPPO + MA-PROPS training on all structurally distinct 2×2 no-conflict matrix games. Solid curves denote means over 100 seeds. Shaded regions denote 95% bootstrap confidence intervals. Takeaway: CoSER reduces sampling error w.r.t. on-policy sampling by a large amount in all games.

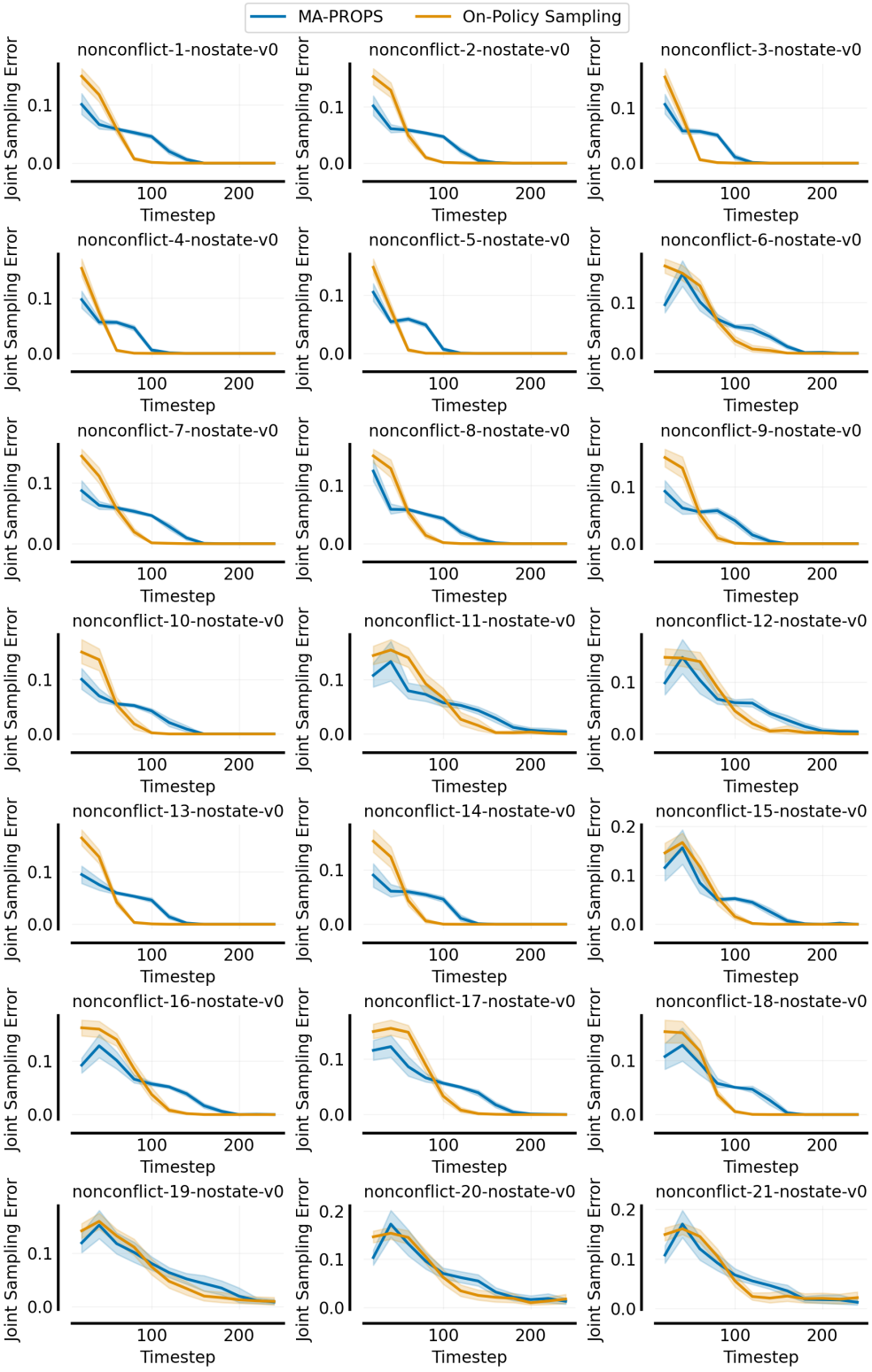


Figure 13: Mean joint sampling error for MAPPO + MA-PROPS training on all structurally distinct 2×2 no-conflict matrix games. Solid curves denote means over 100 seeds. Shaded regions denote 95% bootstrap confidence intervals. Takeaway: Early in training, MA-PROPS reduces sampling error w.r.t. on-policy sampling in all games except 19-21. CoSER reduces sampling error by a larger amount in all games (Fig. 12).

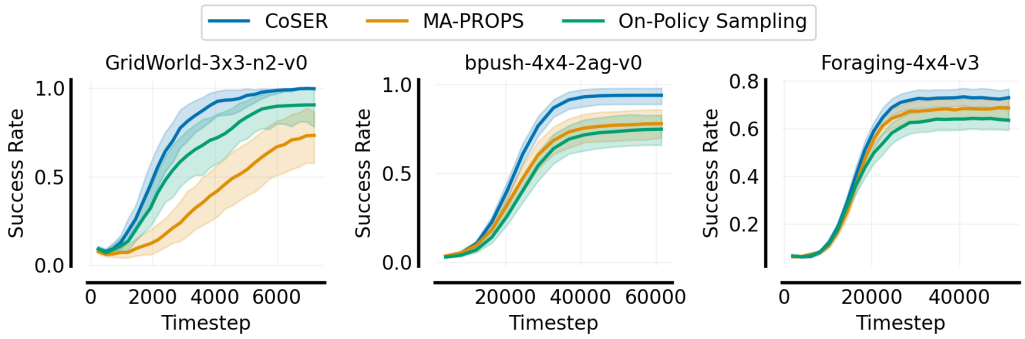


Figure 14: IPPO mean success rate during training for GridWorld, BoulderPush, and LBF. Solid curves denote means over 100 seeds. Shaded regions denote 95% bootstrap confidence intervals.

	2×2 games	3×3 games	GridWorld	BoulderPush	LBF
On-Policy Sampling	3.13	3.21	3.27	3.33	3.47
CoSER	21.62	23.51	25.67	28.98	29.15

Table 1: Time (seconds) required to collect a batch of 2048 samples, averaged over 10 runs.

Behavior learning rate	$0.3, 0.03, 0.003$
Behavior update frequency m	$1, 4$
Behavior KL cutoff	6
Behavior clipping coefficient	$0.3, 1, 10$

Table 2: Hyperparameters used in our hyperparameter sweep for training.

14 Hyperparameters

All hyperparameters used in our experiments are listed in Tables 2, 3, and 4. Our tuning procedure required running approximately one thousand jobs and requires access to many CPUs to do efficiently. We ran all experiments on a computing cluster that enabled us to run several hundred jobs in parallel. The training budget for all experiments is fairly small (less than 100k timesteps), and CoSER and MA-PROPS jobs finish within 1-2 hours. Each job requires 0.6 GB of memory and 1.6 GB of disk.

15 Computational Cost of CoSER

Table 1 compares the time required to collect a batch of 2048 samples using on-policy sampling and CoSER. CoSER is approximately 8 times slower than on-policy sampling, which is expected given that CoSER must update the behavior policy as data is collected. The primary driver of this cost is the number of behavior update epochs, which we fix to 16 following Corrado & Hanna (2023) and do not tune. his work focuses on sample efficiency rather than computational efficiency; we prioritized running many seeds over an extensive hyperparameter sweep.This work focuses on sample efficiency rather than computational efficiency; we prioritized running many seeds over an extensive hyperparameter sweep. It may be possible to reduce the cost without sacrificing sample efficiency by tuning hyperparameters such as the number of behavior update epochs

Batch size	See Table. 4
Learning rate	See Table. 4
Number of update epochs	4
Number of minibatch updates per epoch	4
Discount factor γ	0.99
generalized advantage estimation (GAE)	0.95
Advantage normalization	Yes
Loss clip coefficient	0.2
Entropy coefficient	0.01
Value function coefficient	0.5
Gradient clipping (max gradient norm)	0.5
KL cut-off	None
Actor and critic network architectures	Multi-layer perceptron with hidden layers $(64, 64)$
Optimizer	Adam (Kingma & Ba, 2015)
Number of evaluation episodes	100

Table 3: MAPPO/IPPO hyperparameters across all experiments. We implement MAPPO/IPPO on top of the PPO implementation provided by CleanRL (Huang et al., 2022).

Game	PPO Batch Size	PPO Learning Rate	CoSER/MA-PROPS Learning Rate	CoSER/MA-PROPS Update Frequency
LBF	2048	0.01	0.03	1
BoulderPush	4096	0.003	0.03	1
GridWorld	256	0.01	0.3	1
3×3 matrix games	45	0.1	0.3	1
2×2 matrix games	20	0.1	0.03	1

Table 4: Tuned hyperparameters used in RL training with CoSER and MA-PROPS.