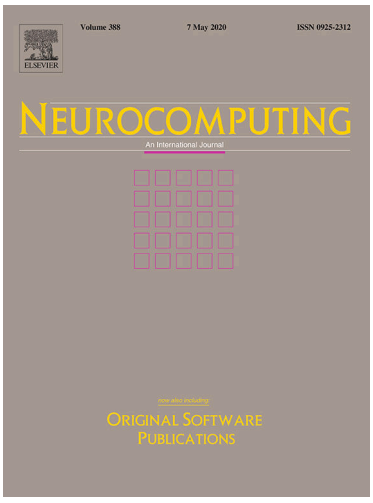
Journal Pre-proofs



PALO Bounds for Reinforcement Learning in Partially Observable Stochastic Games

Roi Ceren, Keyang He, Prashant Doshi, Bikramjit Banerjee

PII: S0925-2312(20)31334-5
DOI: https://doi.org/10.1016/j.neucom.2020.08.054
Reference: NEUCOM 22733

To appear in: *Neurocomputing*

Received Date: 24 December 2019
Accepted Date: 21 August 2020

Please cite this article as: R. Ceren, K. He, P. Doshi, B. Banerjee, PALO Bounds for Reinforcement Learning in Partially Observable Stochastic Games, *Neurocomputing* (2020), doi: https://doi.org/10.1016/j.neucom.2020.08.054

PALO Bounds for Reinforcement Learning in Partially Observable Stochastic Games

Roi Ceren[a], Keyang He[a], Prashant Doshi[a], Bikramjit Banerjee[b]

[a]*THINC Lab, Department of Computer Science*
University of Georgia, Athens, GA 30602-7404 USA
[b]*School of Computing Sciences and Computer Engineering,*
University of Southern Mississippi, Hattiesburg, MS 39406-0001 USA

Abstract

A partially observable stochastic game (POSG) is a general model for multiagent decision making under uncertainty. Perkins' Monte Carlo exploring starts for partially observable Markov decision process (POMDP) (MCES-P) integrates Monte Carlo exploring starts (MCES) into a local search of the policy space to offer an elegant template for model-free reinforcement learning in POSGs. However, multiagent reinforcement learning in POSGs is tremendously more complex than in single agent settings due to the heterogeneity of agents and discrepancy of their goals. In this article, we generalize reinforcement learning under partial observability to self-interested and cooperative multiagent settings under the POSG umbrella. We present *three* new templates for multiagent reinforcement learning in POSGs. MCES for interactive POMDP (MCES-IP) extends MCES-P by maintaining predictions of the other agent's actions based on dynamic beliefs over models. MCES for multiagent POMDP (MCES-MP) generalizes MCES-P to the canonical multiagent POMDP framework, with a single policy mapping joint observations of all agents to joint actions. Finally, MCES for factored-reward multiagent POMDP (MCES-FMP) has each agent individually mapping joint observations to their own action. We use probabilistic approximate locally optimal (PALO) bounds to analyze sample complexity, thereby instantiating these templates to PALO learning. We promote sample efficiency by including a policy space pruning technique and evaluate the approaches on six benchmark domains as well as compare with the state-of-the-art techniques, which demonstrates that MCES-IP and MCES-FMP yield improved policies with fewer samples compared to the previous baselines.

Keywords: multiagent systems, reinforcement learning, POMDP, POSG

1. Introduction

Reinforcement learning (RL) in a multiagent system is a difficult problem, especially in a partially observable setting. A key difficulty is that the agents' strategic interests are

Email addresses: roi.ceren@gmail.com (Roi Ceren), keyang@uga.edu (Keyang He), pdoshi@cs.uga.edu (Prashant Doshi), bikramjit.banerjee@usm.edu (Bikramjit Banerjee)

crucially reliant on the payoff structure of the underlying game, and typically no single algorithm performs best across all types of games. In the past, this has necessitated the concept of "targeted optimality" [1], where a learning algorithm targets certain subsets of scenarios to learn (near)-optimally. However, much of the prior work on multiagent learning has focused on settings that allow agents to observe the global state perfectly, and in some cases, to even observe each others' actions. As real-world applications, such as in robotics, become adept at handling large amounts of observational data that are often noisy and incomplete, it is becoming increasingly imperative to relax these assumptions. Thus in this article, we are interested in partially observable settings where agents do not observe each others' actions perfectly and noisily observe states.

Action-value based reinforcement learning algorithms such as Sarsa [2] and Q-learning [3] represent some of the best performing RL methods in single-agent settings modeled by the Markov decision process (MDP). Sutton and Barto [2] introduced a refinement to Monte Carlo Q-learning allowing the agent to start with a random state-action pair, and called it Monte Carlo Exploring Starts (MCES). If the state is partially observable, the agent may simply use its current observation as the state of a MDP but the learning may not converge. More preferably, the agent conditions its learning on the recent history of observations in a partially observable MDP (POMDP). A T-step policy is a mapping from observation histories of length up to T to an action. Perkins' Monte Carlo exploring starts for POMDP (MCES-P) [4] obtains action-value for a policy from sampled trajectories, and uses it to determine whether a transformed policy in the local neighborhood of the current policy is better. Here, MCES-P transforms the policy at a randomly chosen pair of observation sequence and action to obtain a locally transformed policy. This less explored avenue has the strong benefit of directly searching a discrete space of policies and does not require the convergence of action-values for each state before the policy is obtained. Perkins' elegant and transparent approach serves as the basis for our learning algorithms in multiagent settings presented in this article.

While general multiagent settings are most realistically modeled as partially observable stochastic games (POSGs) [5], it is convenient to consider two popular subcategories under POSGs: self-interested and strictly cooperative settings, the former encompassing adversarial as well as non-adversarial, non-cooperative settings. Cooperative scenarios are usually modeled as decentralized POMDPs [6] and multiagent POMDPs [7] (MPOMDPs can be seen as a type of decentralized POMDP where the agents communicate their observations to each other perfectly), which offer agents opportunities for learning collaboratively, leading to specialized learning algorithms. On the other hand, self-interested settings may lack such opportunities because agents cannot make assumptions about the disposition of other agents. In such cases, agents must act in an individually rational manner, and the learning problem could be approached from an egocentric, self-interested perspective. An interactive POMDP (I-POMDP) [8] is an appropriate model for such scenarios. Consequently, we seek learning algorithms for these two subcategories and their appropriate models.

We introduce three RL templates for settings challenged by partial observability and multiple agents, all of which generalize Perkins' MCES-P algorithm.

- MCES for interactive POMDP (MCES-IP) that aims to learn a policy for a self-interested agent acting and planning in an I-POMDP setting shared with other agents.

- MCES for MPOMDPs (MCES-MP) that generalizes MCES-P to canonical MPOMDPs with a focus on heterogeneous teams. Such teams are comprised of agents with differing rewards cooperating toward a common goal. It learns a single policy mapping joint observations of all agents to joint actions.
- MCES for factored-reward MPOMDPs (MCES-FMP) with each agent individually mapping joint observations to their own action.
- We instantiate these templates with probably approximately locally optimal (PALO) bounds to provide statistical guarantees of ϵ-local optimality that relate with sample complexity. While these guarantees do not relate to global optimality of the learning, nevertheless they provide a useful theoretical footing for RL for POSGs.
- We present and exploit a parameterized policy search space pruning technique, trading statistical guarantees from PALO bounds for a reduction in the computational burden.

We empirically demonstrate using six problem domains that MCES-IP and MCES-FMP using the derived sample complexities arrive at improved local optima relative to relevant baselines and competing methods under similar conditions. Our experiments also demonstrate that MCES-FMP is a significant improvement over MCES-MP for cooperative domains converging to drastically improved policies under similar sample complexities. These new templates and their instantiations not only offer RL with reduced model requirements in the context of distinct POSG settings but they also represent the first methods that relate sample bounds to (local) optimality for RL in popular competitive and cooperative multiagent contexts.

The rest of this article is organized as follows. We discuss the related work in Section 2. Section 3 briefly reviews POSGs and the interactive POMDP framework, both of which serves as background for the new methods. Our main contributions start in Section 4 which presents the MCES-P and MCES-IP templates for self-interested POSGs and their PALO instances. Section 5 then introduces the MCES-MP and MCES-FMP templates along with their PALO instances for POSGs that model heterogeneous teams. To promote efficiency, we show how the policy search space can be pruned in Section 6; this technique can be utilized in conjunction with all the presented RL methods. Our evaluation domains, baselines, experiments, and their results are discussed in Section 7. We conclude this article with some remarks and future directions in Section 8. The appendix gives the proofs of all the theorems mentioned in the article.

A portion of this article was previously published in the conference proceedings of AAMAS 2016 [9]. This article expands on the conference paper in several ways. In addition to an expanded exposition with illustrations, the article includes (i) a key new contribution toward model-free reinforcement learning with sample complexity bounds in cooperative settings in Section 5, which presents two new algorithms that generalize MCES-P, and (ii) associated experiments and analyses on three new problem domains evaluating the performances of the new algorithms in comparison to three new baselines.

2. Related Work

Several approaches exist for optimizing behavior in uncertain environments with or without the presence of other agents. In the single-agent context, partially observable

Markov decision processes (POMDP) [10] consider optimal outcomes to observations given an explicit model of the environment. In multiagent settings, interactive POMDPs (I-POMDP) [8] and decentralized POMDPs (DEC-POMDP) [6] tackle self-interested and cooperative settings, respectively, by considering the effect of other agents and their actions on the state of the multiagent problem [11]. Solution methods have so far predominantly relied on explicit models of the mechanics of the environment and the opponent. Methods for RL in these multiagent settings have been surprisingly few, despite the fact that noisy sensors on robots often present a partially observable problem.

Early model-free approaches for multiagent problems have been explored in both noncooperative and cooperative settings. A Bayes-Adaptive I-POMDP [12] maintains a vector of latent models of environment mechanics and updates its belief over these models online interleaved with actions. Effectively, it casts model-free learning as planning over an infinite space. Along similar lines, Hoang and Low [13] show how a flat Dirichlet multinomial distribution may be utilized to represent the posterior in interactive Bayes-optimal RL by an agent interacting with other self-interested agents. Differing from our context here, the state is assumed to be perfectly observable.

In cooperative contexts, Monte Carlo Q-Alternating (MCQ-Alt) [14] approximates the dynamics of an environment in the presence of another cooperating agent following a fixed policy. After arriving at a locally optimal policy, the agent fixes its own policy and the other agent then learns. The non-learning agents in the first round use a different learning approach to acquire their initial policies [15]. Regardless of the qualitiy of these initial policies, the alternating best-reponse learning of MCQ-Alt converges to near-optimal policies in benchmark domains. However, in addition to requiring turn-taking, the experiences of the non-learning agents are wasted in this approach. Another related approach overcomes these limitations by allowing agents to learn simultaneously [16], but assumes that agents can perfectly observe hidden information during the training phase. In contrast, our algorithms do not have any of these limitations.

In the context of MPOMDPs, a model-based RL technique which offers the previous best performance is the Bayes-adaptive factored-value POMCP (BA-FV-POMCP) [17]. This method interleaves Bayesian model building with POMCP-based planning to solve MPOMDPs. The model is built using transition and observation count vectors and updated as additional transitions and observations are explored. The learned model is utilized at the beginning of an episode to build the simulator required by the POMCP rollouts. While potentially scalable to large MPOMDPs, POMCP's performance is contingent on the veracity of the learned model. But, count data based models often require a very large number of samples for reasonable accuracy in complex domains as our experiments demonstrate.

Distributed gradient descent (DGD) method [5] performs the same gradient-descent algorithm in parallel for each agent's local policy. The policy learning and control is distributed among independent agents who are not aware of each others' actions as long as agents learn simultaneously. DGD is only guaranteed to find local optima in the space of factored policies, however, it will not always converge to a Nash equilibrium.

A deep learning based approach for multi-task multiagent reinforcement learning under partial observability [18] conducts single task specialization, and subsequently unifies task-specific deep recurrent Q-Nets (DRQNs) into a joint policy that performs well in all tasks. It combines hysteretic learners, DRQNs, concurrent experience replay

trajectories (CERTs), and distillation. However, the training process requires a very large number of epochs (each epoch entails a simulation of the problem), about 10^5 even for a 2-agent toy problem.

Another deep learning algorithm is the multiagent deep deterministic policy gradient (MADDPG) [19]. It is a multiagent actor-critic method that utilizes a decentralized actor and a centralized critic with extra information about the policies of other agents. The assumption of knowing other agents' policies can be relaxed by inferring policies of other agents using maximum likelihood estimation on other agents' actions. A more recent deep multiagent reinforcement learning algorithm, called "learning with opponent learning awareness" (LOLA) [20], was designed to handle non-stationarity arising naturally in multiagent learning, by allowing agents to observe (or estimate via maximum likelihood opponent modeling) the network parameters of other agents. In particular, a LOLA agent explicitly accounts for how its change of parameters will affect the value function of other agents and incorporates the effect in its own value function. Thus a LOLA agent is able to actively influence the future policy updates of other LOLA agents. However, the performance of the opponent modeling version was significantly worse. Furthermore, both MADDPG and LOLA were only evaluated in repeated games although they may be applied to POSGs. RL from hierarchical critics [21] is another deep multiagent RL method. Every agent has its own local critic, and there is also a global critic. Each agent passes its own value function to the learning manager, which compares all agents' value functions and the global critic's value function, then returns maximum value function to each agent for advantage calculation. This method can only be used in cooperative setting, and agents need to share their value functions.

Egorov et al. [22] discusses three neural network based approaches to multiagent cooperative learning. While the centralized approach utilizes a centralized learner to manage learning, the concurrent method allocates a neural net to each agent but with no coordination among them and learns local policies. However, the best performing approach adds parameter sharing between these concurrent nets thereby facilitating coordination through sharing of experiences. The latter method scales to large multiagent control tasks with dozens of agents through curriculum learning, which adds a considerable amount of training time depending on the complexity of the problem domain. Curriculum learning is modeled on the idea that learning benefits when samples are presented in order of increasing difficulty. It is difficult to define a curriculum for many complex problems, and unrealistic for many real-world applications due to the very large sample requirements. These sample-intensive approaches contrast with our orthogonal focus on rigorous sample bounds for RL that is cognizant of the fact that sample generation in multiagent systems is not easy. Furthermore, agents in these approaches are homogeneous and tasks for a heterogeneous team of agents is identified as an avenue of future work in these approaches.

3. Background

In this section, we discuss the most relevant background for our proposed approach. We first introduce a general model of multiagent interactions, POSGs, which encompasses all special cases addressed in this paper. Then, we briefly review the well-known I-POMDP framework for modeling self-interested interactions in Section 3.2.

3.1. Partially Observable Stochastic Games

A partially observable stochastic game (POSG) is a general model of multiagent interactions under noise, uncertainty, and incomplete observations. Formally, a POSG is a tuple $\langle \mathcal{I}, S, A, \Omega, T, O, \mathcal{R} \rangle$, where,

- $\mathcal{I} = \{1, ..., Z\}$ is the set of Z agents;
- S is the set of states;
- $A = A_1 \times ... \times A_n$, is the set of joint actions where A_1, A_2, ..., A_Z are the sets of the individual agent's actions. A joint action is then, $\bar{a} = \langle a_1, ..., a_Z \rangle$;
- $\Omega = \Omega_1 \times ... \times \Omega_Z$, is the set of joint observations where Ω_1, Ω_2, ..., Ω_Z are the sets of individual agent's observations. A joint observation is $\bar{o} = \langle o_1, ..., o_Z \rangle$;
- $T : S \times A \times S \mapsto [0, 1]$ is the transition function that determines how the state evolves. It maps an origin state, a joint action, and an arrival state to a probability;
- $O : S \times A \times \Omega \mapsto [0, 1]$ is the observation function that gives the informativeness of the observations toward the state. It maps an observation, the arrival state, and a joint action to a probability;
- $\mathcal{R}_i : S \times A \mapsto \mathbb{R}$ is the reward function of the ith agent, $i = 1, \ldots, Z$.

A game can have finite or infinite stages, where the number of stages is called the horizon of the game. At each stage, all agents simultaneously select an action and receive a reward and an observation, transitioning to the next state (following transition function T) that defines the next state of the game. The objective of a self-interested agent is to maximize the expected sum of rewards it receives over the horizon of the game. Whether agents compete or cooperate in seeking reward depends on their reward functions. For instance, purely competitive settings are characterized by reward functions where $\sum_i \mathcal{R}_i(s, \vec{a})$ is a constant for all $s, \vec{a}$. That is, an agent's reward cannot be improved without reducing another agent's reward. On the other hand, purely cooperative settings have $\mathcal{R}_1(s, \vec{a}) = \mathcal{R}_2(s, \vec{a}) = \ldots = \mathcal{R}_n(s, \vec{a})$ for all $s, \vec{a}$, i.e., all agents' rewards coincide leading to identical interest for the agents. A POMDP is a special case of a POSG with a *single* controller, i.e., a single-agent POSG, which retains the characteristic of a decision process under uncertainty, but is no longer a game.

3.2. Interactive POMDPs

Interactive POMDPs [8] generalize POMDPs to self-interested multiagent settings. They model the subjective perspective of a self-interested agent situated in a POSG. In a setting shared by two intentional agents i and j, agent i's I-POMDP with l levels of nesting is defined by the tuple:

$$\text{I-POMDP}_{i,l} \triangleq \langle IS_{i,l}, A, T_i, \Omega_i, O_i, R_i \rangle$$

- $IS_{i,l}$ is the set of interactive states defined as, $IS_{i,l} = S \times M_{j,l-1}$, where $M_{j,l-1} = \{\Theta_{j,l-1} \cup SM_j\}$, for $l \geq 1$, and $IS_{i,0} = S$, where S is the set of physical states. $\Theta_{j,l-1}$ is the set of computable, intentional models ascribed to agent j : $\theta_{j,l-1} = \langle b_{j,l-1}, \hat{\theta}_{j,l-1} \rangle$, where $b_{j,l-1}$ is agent j's level $l-1$ belief, $b_{j,l-1} \in \triangle(IS_{j,l-1})$, and $\hat{\theta}_{j,l-1} = \langle A, T_j, \Omega_j, O_j, R_j \rangle$, is j's frame. Here, j is assumed to be Bayes-rational.

- $A = A_i \times A_j$ is the set of joint actions of all agents.
- $T_i : S \times A \times S \mapsto [0, 1]$ is the transition model.
- Ω_i is the set of observations for agent i.
- $O_i : S \times A \times \Omega_i \mapsto [0, 1]$ is the observation function.
- $R_i : S \times A \mapsto \mathbb{R}$ is the reward function for agent i.

Agent i's belief over its interactive states is a sufficient statistic for its observation history. Generalizing the belief update from POMDP to I-POMDP is not trivial because physical state transitions depend on actions performed by both agents. The prediction of the transition has to be made based on the probabilities of various actions of the other agent. In order to predict the next physical state, i must update its beliefs about j's model based on its anticipation of how j updates its belief. Agent i's belief update at time step t is defined by:

$$b^t_{i,l}(i, s^t) = \beta \sum\nolimits_{is^{t-1}} b^{t-1}_{i,l}(is^{t-1}) \sum\nolimits_{a^{t-1}_j} P(a^{t-1}_j|\theta^{t-1}_{j,l-1}) T_i(s^{t-1}, a^{t-1}, s^t) \\ \times O_i(s^t, a^{t-1}, z^t_i) \sum\nolimits_{z^t_j} P(b^t_{j,l-1}|b^{t-1}_{j,l-1}, a^{t-1}_j, z^t_j) O_j(s^t, a^{t-1}, z^t_j) \quad (1)$$

where β is a normalizing factor and $P(a^{t-1}_j|\theta^{t-1}_{j,l-1})$ is the probability that a^{t-1}_j is Bayes-rational for an agent modeled by $\theta^{t-1}_{j,l-1}$.

Given the generalized belief update, solution to an I-POMDP is a policy, analogous to a POMDP. Using the Bellman equation, each belief state in an I-POMDP has a value which is the maximum reward the agent can expect starting from that belief state and over the future. Agent i's optimal action for a belief state is an element of the set of actions that optimize the value at that belief for a finite or infinite horizon.

4. Self-Interested RL in POSGs

A key measure of performance for an RL agent is the number of environment interactions needed before its behavior policy meets certain performance criteria. This measure is often referred to as *sample complexity*. Our goal is to design model-free RL algorithms with performance guarantees in terms of sample complexity.

The literature on theoretically well-founded, model-free RL in partially observable multiagent settings is relatively sparse compared to single agent settings. In our review of the related literature, we did not find any algorithm for model-free RL that specifically targets POSGs with self-interested agents, although several algorithms exist that can be trivially extended to POSGs by considering states as observations. These algorithms, discussed in Section 2, are often designed for (or evaluated in) fully observable or repeated game settings. They are unlikely to perform well when extended to POSGs, since partial observability introduces the need to either maintain observation histories or explicitly incorporate and update beliefs. Both of these choices introduce additional complexity as well as impact performance, but they are omitted in such algorithms. The infinite regional policy representation (iRPR) [23] performs model-free exploration of nonparametric policies for POMDPs. While iRPR allows an unbounded number of states, its convergence is sample-intensive requiring 10^3 samples even for the simple

1D-maze domain; this makes it a poor departure point. Additionally, parameters must be manually configured to achieve optima.

Perkins' MCES-P [4] offers a template for online model-free RL in single agent settings in a way that differs substantially from traditional model-free methods such as Q-learning. At its core, it hill climbs the space of neighboring policies, which draws comparisons to policy iteration rather than the value iteration of Q-learning. Perkins also notes a particular instantiation of MCES-P that utilizes Hoeffding's inequality to ensure that a sufficient number of samples are taken so that a given difference in action-value is observed with at least some probability. This instantiation is theoretically founded on Greiner's probably approximately locally optimal (PALO) learning system [24]. Subsequently, MCES-P offers an appealing template for generalizing to multiagent settings with the potential to fill this wide gap. While we initially focus on learning by *self-interested* agents situated in a multiagent setting in this section, MCES-P is sufficiently generic for an extension to learning the joint policies of multiple agents in cooperative settings as well, which we discuss in Section 5.

We begin by discussing the consequence of a straightforward and naive extension of MCES-P to self-interested multiagent settings. This is followed by a more sophisticated generalization that models the other interacting agents.

4.1. A Naive Extension of MCES-P

Perkins' MCES-P template [4] may be utilized almost as is in a multiagent setting. Therefore, we begin by exploring this approach followed by presenting a new generalization of MCES-P that improves on it significantly. Toward this end, we reproduce Perkins' algorithm in Algorithm 1. A random observation sequence, $\vec{o}$, and the corresponding action are picked, then the latter replaces the previous action at $\vec{o}$ thereby transforming the policy. [1] Let π_i be the initial seed policy and $\pi_i \leftarrow (\vec{o}_i, a_i)$ be this policy but transformed to perform action a_i on observing $\vec{o}_i$; the latter denotes a neighboring policy. Q-values are maintained to estimate the values of the policies, with and without this transformation. Since the values of these policies only differ in the reward sequence following the observation of $\vec{o}_i$, the sum of rewards from only this part of a trajectory τ, denoted as $R_{post-\vec{o}_i}(\tau)$, contribute to Q (see line 7 of Algorithm 1). Parameter α is an averaging learning rate, $\alpha(m, c) = \frac{1}{c+1}$, where m is the number of transformations taken so far and c the count of updates to the Q-function. In this instance, m does not affect the learning rate. Subsequently, MCES-P proceeds by randomly picking an observation sequence at which to transform a policy, sufficiently simulating the original and transformed policies and updating their Q-values with new information for comparison. The transformed policy is adopted if its Q-value exceeds that of the original by ϵ both updated across k samples. The algorithm terminates in the absence of policy transformations for some time.

In instantiating the template for PALO bounds, we assume that the other agent is guided by a fixed policy or a fixed distribution over policies (i.e., mixed strategy). This

[1] Perkins' MCES-P curbs the policies to actions contingent on a single observation, ie., memory-less policies. Therefore, it initially picks a single observation and action only. We extend the algorithm here to consider a *sequence* of observations for better performance.

Algorithm 1 MCES-P in Multiagent Settings

Require: Q-value table initialized to all 0s; initial policy π_i that is greedy w.r.t. Q-values; learning rate schedule α; horizon T; and error ϵ

$c_{\vec{o}_i,a_i} \leftarrow 0$ for all $\vec{o}_i$ and a_i
$m \leftarrow 0$
repeat
  Pick some observation history $\vec{o}_i$ and action a_i
  Modify π_i to $\pi_i \leftarrow (\vec{o}_i, a_i)$
  Generate trajectory τ of length T in the multiagent setting according to $\pi_i \leftarrow (\vec{o}_i, a_i)$ (this involves simulating the policies of other agents as well)
  $Q_{\pi_i \leftarrow \vec{o}_i,a_i} \leftarrow (1-\alpha(m, c_{\vec{o}_i,a_i}))\, Q_{\pi_i \leftarrow \vec{o}_i,a_i} + \alpha(m, c_{\vec{o}_i,a_i})\, R_{post-\vec{o}_i}(\tau)$
  $c_{\vec{o}_i,a_i} \leftarrow c_{\vec{o}_i,a_i} + 1$
  if $\max_{a_i'} Q_{\pi_i \leftarrow \vec{o}_i,a_i'} - Q_{\pi_i} > \epsilon(m, c_{\vec{o}_i,a_i}, c_{\vec{o}_i,\pi_i(\vec{o}_i)})$ **then**
    $\pi_i(\vec{o}_i) \leftarrow a_i'$ where $a_i' \in \arg\max Q_{\pi_i \leftarrow \vec{o}_i,a_i'}$
    $m \leftarrow m + 1$
    for all $\vec{o}_i, a_i$ **do**
      $c_{\vec{o}_i,a_i} \leftarrow 0$
until termination

ensures that the sampling distribution is fixed and Hoeffding's inequality continues to apply. We point out that this assumption differentiates our problem from the traditional multiagent RL [25] where all agents are learning simultaneously and therefore the learning problem is not stationary.

Given the above, MCES-P-PALO in multiagent settings may face the same four types of errors due to sampling as those faced in single agent settings (see proof of Proposition 1 in Appendix). However, the error due to sampling and the sample complexity differ because of the presence of other agents. The size of the policy neighborhood N of agent i increases to $\mathcal{O}(|A|\frac{|\Omega|^T-1}{|\Omega|-1})$ due to our consideration of the observation history. Let $\Lambda(\pi_i, \pi_i')$ be the upper bound on the range of the difference between the action-values of π_i and some other policy π_i'. For simplicity of presentation, we consider one other agent j in the environment. Let $R_{i,max}$ now be defined as, $R_{i,max} \triangleq \max_{s,a_i,a_j} R_i(s,a_i,a_j)$ and analogously for $R_{i,min} \triangleq \min_{s,a_i,a_j} R_i(s,a_i,a_j)$. Let $\tilde{\mathcal{T}}$ be a set of sampled trajectories. Then, we get,

$$\begin{aligned}\Lambda(\pi_i,\pi_i') &\triangleq \max_{\tilde{\mathcal{T}}} (Q_{\pi_i} - Q_{\pi_i'}) - \min_{\tilde{\mathcal{T}}} (Q_{\pi_i} - Q_{\pi_i'}) \\ &\leq \sum_{t=0}^{T-1} \{(R_{i,max} - R_{i,min}) - (R_{i,min} - R_{i,max})\} \\ &= \sum_{t=0}^{T-1} 2(R_{i,max} - R_{i,min}) = 2T(R_{i,max} - R_{i,min}) \qquad (2)\end{aligned}$$

Consequently, the threshold for comparison at stage m becomes:

$$\epsilon(m,p,q) = \begin{cases} \Lambda(\pi_i,\pi_i')\sqrt{\frac{1}{2p}\ln\frac{2(k_m-1)N}{\delta_m}} & \text{if } p=q<k_m \\ \frac{\epsilon}{2} & \text{if } p=q\geq k_m \\ +\infty & \text{otherwise} \end{cases} \qquad (3)$$

and the sample complexity is:

$$k_m \leftarrow \left\lceil 2\left(\frac{\Lambda}{\epsilon}\right)^2 \ln \frac{2N}{\delta_m} \right\rceil \tag{4}$$

while the probability δ_m remains the same as in Section 3. Notice that in so called "neutral settings" where agent i's reward does not depend on j's action (although the state is still impacted by j's action) $R_{i,max}$ and $R_{i,min}$ collapse into R_{max} and R_{min}, respectively causing no change in the PALO bounds from Perkins [4]. However, in competitive settings $R_{i,max}$ is often greater than R_{max} whereas $R_{i,min}$ tends to be smaller than its counterpart due to which the sample complexity is higher or the error is greater for the same number of samples.

MCES-P-PALO terminates if there has been no policy change when k_m samples are reached or if no neighboring policy exceeds the current policy in action value by more than $\epsilon - \Lambda(\pi_i, \pi_i') \cdot \sqrt{\frac{1}{2p}\ln\frac{2(k_m-1)N}{\delta_m}}$ for lesser samples.

Proposition 1 shows that MCES-P-PALO in partially observable multiagent settings will terminate and converge to an ϵ-locally optimal policy.

Proposition 1 (Local optimality of MCES-P-PALO).
Instantiation MCES-P-PALO incrementally produces a series of policies π_i^1, π_i^2, ..., π_i^n, such that each π_i^{n+1} is a local neighbor of π_i^n and with probability at least $1-\delta$:

1. *Each policy π_i^{n+1} has an expected value strictly greater than its predecessor, π_i^m where $1 \leq n \leq m-1$;*

2. *Final policy π_i^m returned by MCES-P-PALO is ϵ-locally optimal such that there is no neighbor of π_i^m given our transformation procedure whose expected value exceeds that of π_i^m by more than ϵ.*

Moreover, MCES-P-PALO will terminate with probability 1 if N is finite.

The full proof of this proposition is given in the Appendix. It presents four types of errors that are possible due to sampling. As part of the proof, the expressions for $\epsilon(m, p, q)$, k_m and δ_m given previously are also established.

4.2. Modeling Other Agents: MCES-IP

While Proposition 1 is appealing, a limitation of MCES-P-PALO is that the required sample size is very large. For example, for an ϵ of 0.05 and probability δ = 0.1 the required sample size k_m is 320,200 for the small two-agent competitive Tiger problem [8]. Can we significantly reduce this alarming sample complexity and if so what is the trade-off?

A key insight may allow us to mitigate the sample complexity: *If we can predict the other agent's actions, then we may simply optimize in that specific context.* Toward this, we partially relax the model-free characteristic of the RL to obtain savings in samples. We may divide agent i's observations into those that are public and those that are private. The former type is public signals that are shared between agents while the latter are signals privately observed by an agent. This division is without loss of

generality because the set of public observations may be empty if agents have private observations only and the set of private observations is empty if only public observations are obtained.

Next, let the private monitoring at time t convey information to the agent about the other agent's action at $t-1$ albeit noisily while the public signal provides uncertain information about the common state of the system. We argue that this specificity is often seen in multiagent problems. For example, agents in the multiagent Tiger problem hear growls that inform about the location of the tiger and each agent may also hear a creak from the left or right, or do not hear a creak indicating the door, if any, that was opened by the other agent. As a final step, we depart from the completely model-free setting of MCES-P and let a joint private observation function that models the information content of private observations only be common knowledge. The agents are not aware of any other model parameters.

Given the setup above, an agent in the MCES for interactive POMDP (MCES-IP) template starts with a prior distribution over possible models of the other agent and updates the prior as it receives private observations. As the agent acts and observes, a sequence of beliefs are obtained and the agent utilizes both sequences: observation sequence and belief sequence to decide on an action. We show the MCES-IP template in Algorithm 2.

Algorithm 2 MCES-IP

Require: Q-value table initialized to all 0s; initial policy, π_i, that is greedy w.r.t. Q-values; prior on set of models M_j; learning rate schedule α; horizon T; and error ϵ

1: $c^{\vec{a}_j}_{\vec{o}_i,a_i} \leftarrow 0$
2: $m \leftarrow 0$
3: **repeat**
4: Pick some observation history, $\vec{o}_i$, and a_i
5: Modify π_i to $\pi_i \leftarrow (\vec{o}_i, a_i)$
6: Generate trajectory τ of length T according to $\pi_i \leftarrow (\vec{o}_i, a_i)$ (requires simulating other agents' policies)
7: Generate belief sequence $\vec{b}_i$ based on τ using Eq. 5
8: Obtain most probable action sequence $\vec{a}_j$ from $\vec{b}_i$
9: $Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i,a_i} \leftarrow (1-\alpha(m, c^{\vec{a}_j}_{\vec{o}_i,a_i})) \cdot Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i,a_i} + \alpha(m, c^{\vec{a}_j}_{\vec{o}_i,a_i}) \cdot R_{post-\vec{o}_i}(\tau)$
10: $c^{\vec{a}_j}_{\vec{o}_i,a_i} \leftarrow c^{\vec{a}_j}_{\vec{o}_i,a_i} + 1$
11: **if** $\max_{a'_i} Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i,a'_i} - Q^{\vec{a}_j}_{\pi_i} > \epsilon^{\vec{a}_j}(m, c^{\vec{a}_j}_{\vec{o}_i,a_i}, c^{\vec{a}_j}_{\vec{o}_i,\pi_i(\vec{o}_i)})$ **then**
12: $\pi_i(\vec{o}_i) \leftarrow a'_i$ where $a'_i \in \arg\max_{a'_i} Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i,a'_i}$
13: $m \leftarrow m+1$
14: **for all** $\vec{o}_i, a_i, \vec{a}_j$ **do**
15: $c^{\vec{a}_j}_{\vec{o}_i,a_i} \leftarrow 0$
16: **until** termination

MCES-IP generally follows the procedure of MCES-P. It builds on the latter by additionally generating a belief sequence $\vec{b}_i$ using the actions and observations in a trajectory (line 7); each belief is a distribution over a pre-defined set of models of the other agent. As the space of possible belief sequences is continuous, MCES-IP picks

the most-probable model from each belief and the corresponding predicted action, to obtain a corresponding action sequence $\vec{a}_j$ (line 8). Q-values and update counts are now indexed using this action sequence (lines 9-10). In this way, action values of policies are specific to predicted actions of the other agent, which allow a more informed probabilistic hill climbing in multiagent settings.

Belief sequence, $\vec{b}_i$, is generated as follows. A trajectory τ_i is $\{a_i^0, r_i^0, \langle o^1, \omega_i^1\rangle, a_i^1, r_i^1, \langle o^2, \omega_i^2\rangle, a_i^2, r_i^2, \ldots, \langle o^{T-1}, \omega_i^{T-1}\rangle, a_i^{T-1}, r_i^{T-1}\}$. Notice that each observation in the trajectory is composed of public and private signals; denote $\vec{o}_i \triangleq \langle \vec{o}, \vec{\omega}_i\rangle$. Let M_j^t be a discrete set of j's models at time step t, where $m_j^t \in M_j^t$ is: $m_j^t \triangleq \langle h_j^t, \pi_j\rangle$, h_j^t is j's action-observation history of length t which when given as input to j's policy π_j produces the predicted action at time t. Agent i's belief b_i is a distribution over M_j updated based on i's action and observation as given below:

$$b_i'(m_j^{t+1}|a_i^t, o^{t+1}, \omega_i^{t+1}, b_i) = \sum\nolimits_{m_j^t \in M_j^t} b_i(m_j^t) \sum\nolimits_{a_j^t \in A_j} Pr(a_j^t|m_j^t) \\ \times O_i(\omega_i^{t+1}|a_i^t, a_j^t)\ \delta_K(h_j^{t+1}, \text{APPEND}(h_j^t, a_j^t, o^{t+1})) \tag{5}$$

As a part of updating its belief, i must first update its models of j and in particular, the action-observation history contained in each model using the predicted action a_j^t and public observation o^{t+1}; this is performed by APPEND. Kronecker delta function, δ_K, is 1 if an updated model matches the one in m_j^{t+1} otherwise it is 0. Private observation function O_i is the marginal of the joint, and it allows using the private signal to weight predicted actions and by backward inference the models that generated the actions. This likelihood is then propagated forward to the updated model, m_j^{t+1}. As such, Eq. 5 is a sophisticated Bayesian belief update.

A sequence of beliefs $\vec{b}_i$ is then generated by updating the uniform prior with the action-observation pairs in a trajectory using Eq. 5; thus the length of this sequence is T. We may pick the most probable model from each belief in the sequence, $\arg\max_{m_j} b_i(m_j)$, and get the model-predicted action, $\arg\max_{a_j} Pr(a_j|m_j)$, to obtain the action sequence $\vec{a}_j$.

We instantiate Algorithm 2 to obtain MCES-IP with PALO bounds, which we denote as MCES-IP-PALO. For this, we assume that the error is due to sampling trajectories only and that the monitoring is perfect, i.e., private signals perfectly reveal j's action. We discuss the effect on the bounds due to observation noise later in this section.

For a given error ϵ and probability δ let,

$$\epsilon^{\vec{a}_j}(m,p,q) = \begin{cases} \Lambda^{\vec{a}_j}(\pi_i, \pi_i')\sqrt{\frac{1}{2p}\ln\frac{2(k_m-1)N}{\delta_m}} & \text{if } p = q < k_m \\ \frac{\epsilon}{2} & \text{if } p = q \geq k_m \\ +\infty & \text{otherwise} \end{cases} \tag{6}$$

where

$$k_m = \left\lceil 2\frac{(\Lambda^{\vec{a}_j}(\pi_i, \pi_i'))^2}{\epsilon^2}\ln\frac{2N}{\delta_m}\right\rceil \qquad and \qquad \delta_m = \frac{6\delta}{m^2\pi^2} \tag{7}$$

Here, $\Lambda^{\vec{a}_j}(\pi_i, \pi_i')$ is an upper bound on the range of the difference in action-values between two policies given j's action sequence is $\vec{a}_j$. Note that k_m is polynomial in

problem parameters, since $\ln(N)$ is linear in T, the problem horizon. This leads to a *polynomial sample complexity per transform*. Let $R^{a_j}_{i,max} = \max_{s,a_i} R_i(s, a_i, a_j)$ and analogously for $R^{a_j}_{i,min}$; these specific values are assumed to be known. Then, we get:

$$\begin{aligned}\Lambda^{\vec{a}_j}(\pi_i, \pi'_i) &= \max_{\tilde{\mathcal{T}}} \left(Q^{\vec{a}_j}_{\pi_i} - Q^{\vec{a}_j}_{\pi'_i}\right) - \min_{\tilde{\mathcal{T}}} \left(Q^{\vec{a}_j}_{\pi_i} - Q^{\vec{a}_j}_{\pi'_i}\right) \\ &\leq \sum_{t\in T} \left(R^{a^t_j}_{i,max} - R^{a^t_j}_{i,min}\right) - \left(R^{a^t_j}_{i,min} - R^{a^t_j}_{i,max}\right) \\ &= \sum_{t\in T} 2\left(R^{a^t_j}_{i,max} - R^{a^t_j}_{i,min}\right) \end{aligned} \tag{8}$$

The following key proposition indicates the benefit of predicting the other agent's actions on sample complexity albeit at the expense of the belief update run time.

Proposition 2 (Reduced sample complexity). *For any predicted action sequence, $\vec{a}_j$,*

$$\Lambda^{\vec{a}_j}(\pi_i, \pi'_i) \leq \Lambda(\pi_i, \pi'_i)$$

where Eqs. 2 and 8 define $\Lambda(\pi_i, \pi'_i)$ and $\Lambda^{\vec{a}_j}(\pi_i, \pi'_i)$, respectively. The proof of this proposition is straightforward and is given in the Appendix. Subsequently, Proposition 2 entails that the sample size bound k_m for MCES-IP-PALO is also less than or equal to the corresponding sample size bound for MCES-P-PALO. The effect is significant because k_m grows quadratically with Λ. On the other hand, the Q-values table for MCES-IP-PALO expands significantly with up to $\frac{|\Omega|^T-1}{|\Omega|-1}$ values for each i's policy or its transformation. As such, the reduced sample bound of MCES-IP-PALO must be less by a factor of $\frac{|\Omega|^T-1}{|\Omega|-1}$ in the worst case to be effective. This is often the case as we demonstrate for the two-agent Tiger problem where k_m for MCES-IP-PALO is as low as 106,822 that is almost three times less than that for MCES-P-PALO.

If private signals provide perfect information about j's actions, then MCES-IP-PALO terminates when,

$$Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i, a'_i} < Q^{\vec{a}_j}_{\pi_i} + \epsilon - \epsilon^{\vec{a}_j}(m, c^{\vec{a}_j}_{\vec{o}_i, a_i}, c^{\vec{a}_j}_{\vec{o}_i, \pi_i(\vec{o}_i)})$$

for all $\vec{o}_i$, $a'_i \neq \pi_i(\vec{o}_i)$, and we have encountered at most $\frac{|\Omega|^T-1}{|\Omega|-1}$ many distinct $\vec{a}_j$ in the trajectories for each $\vec{o}_i$, a'_i pair. Under the same assumption of perfect monitoring, for the comparison threshold, sample bound and probability as defined above, we obtain the following theorem for MCES-IP-PALO.

Proposition 3 (Local optimality under perfect monitoring). *Template MCES-IP-PALO under perfect monitoring incrementally produces a series of policies π^1_i, π^2_i, ..., π^m_i, such that each π^{q+1}_i is a local neighbor of π^q_i and with probability at least $1-\delta$:*

1. *Each policy π^{q+1}_i has an expected value strictly greater than its predecessor, π^q_i where $1 \leq q \leq m-1$;*

2. *Final policy, π_i^m returned by* MCES-IP-PALO *is ϵ-locally optimal such that there is no neighbor of π_i^m given our transformation procedure whose expected value exceeds that of π_i^m by more than ϵ.*

Moreover, MCES-IP-PALO *terminates with probability 1 if N is finite.*

The proof of this proposition proceeds analogously to the proof for Proposition 1, with Λ replaced with $\Lambda^{\vec{a}_j}$.

We can generalize the above results on MCES-IP-PALO for the case of *imperfect monitoring* – when the probability of error in estimating $\vec{a}_j$ is known, say δ_e. In this case, the agent may place Q-samples in the wrong $Q^{\vec{a}_j}$ bins (see line 9 of Algorithm 2), leading to non-identically distributed samples in a bin. Fortunately, given δ_e we may generalize Proposition 3 to the case of independent but non-identically distributed samples using a more general form of Hoeffding's inequality. Analogously to Eq. 8, let $\bar{\Lambda}^{\vec{a}_j}$ be an upper bound on the range of differences in action-values for all j's action sequences that are *different* from $\vec{a}_j$. Then for the case of $p = q < k_m$, we redefine $\epsilon^{\vec{a}_j}(m, p, q)$ as,

$$\epsilon^{\vec{a}_j}(m, p, q) = \sqrt{(1-\delta_e)(\Lambda^{\vec{a}_j})^2 + \delta_e(\bar{\Lambda}^{\vec{a}_j})^2}\sqrt{\frac{1}{2p}\ln\frac{2(k_m-1)N}{\delta_m}} \tag{9}$$

where k_m is redefined as

$$k_m = \left\lceil \frac{2((1-\delta_e)(\Lambda^{\vec{a}_j})^2 + \delta_e(\bar{\Lambda}^{\vec{a}_j})^2)}{\epsilon^2}\ln\frac{2N}{\delta_m} \right\rceil \tag{10}$$

Algorithm 2 requires a slight modification for this case. For convenience, let $\zeta^{\vec{a}_j} = \max_{a_i'} Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i, a_i'} - Q^{\vec{a}_j}_{\pi_i}$. Then, line 11 of Algorithm 2 changes to the following:

$$(1-\delta_e)\zeta^{\vec{a}_j} + \delta_e\bar{\zeta}^{\vec{a}_j'} > (1-\delta_e)\epsilon^{\vec{a}_j} + \delta_e\bar{\epsilon}^{\vec{a}_j'}$$

The implicit assumption above is that when $Q^{\vec{a}_j}$ receives a wrong sample meant for bin $\vec{a}_j'$, the action sequence is equally likely to be any $\vec{a}_j' \neq \vec{a}_j$. Therefore, $\bar{\zeta}^{\vec{a}_j'}$ is the mean of $\zeta^{\vec{a}_j'}$ for all $\vec{a}_j' \neq \vec{a}_j$ seen so far, and analogously $\bar{\epsilon}^{\vec{a}_j'}$ is also the mean. Notice that insisting on the test of line 11 for every $\vec{a}_j$ before the current policy is changed, would be a stronger form of this test, and hence also sufficient. Finally, we note that when $\delta_e = 0$, we recover MCES-IP-PALO for perfect monitoring as a special case of this setting.

Clearly, the above approach is only useful if we can estimate the error due to imperfect monitoring, δ_e. Direct comparison with the perfect monitoring case suggests that one possible way to estimate it could exploit domain knowledge, if available, of how much the private observations are decorrelated from the other agents' actions.

5. RL for Cooperative POSGs

Next, we turn our attention to partially observable multiagent settings where the agents must coordinate to achieve the maximum reward. Such settings may be viewed as

POSGs with a single (shared) reward function for the collection of agents (or analogously, each agent having the same reward function as others). We introduce two model-free RL approaches for cooperative settings. Both generalize the MCES-P template.

5.1. Heterogeneous team

We further focus on a system of n heterogeneous agents cooperating toward a common goal. Specifically, $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$ is the collection of agent's reward functions, which may differ; this defines the heterogeneity. Here, $\mathcal{R}_i : S \times A \mapsto \mathbb{R}$, $i \in \mathcal{I}$ is an agent's reward function, which maps state and joint action to value. An agent's reward $\mathcal{R}_i$ is decomposed into local costs (R_i), predicated on the joint physical state and *individual* action, and the global reward (R_G), a mapping from the joint state and joint action, as shown below.

$$\mathcal{R}_i(s, \vec{a}) = R_i(s, a_i) + R_G(s, \vec{a}) \tag{11}$$

Local costs may be unique to each agent and provide a way to model the diversity between agents in *heterogeneous* teams. For example, the global reward in the multi-robot alignment problem illustrated in Fig. 3 is 100 when the robots are aligned, otherwise -1. The local component differs between robots with the more sophisticated platform incurring a cost of 2 for turning and 1 for emitting infrared and a higher probability to make a successful turn while it costs 3 for the other robot to do either and a lower probability to make a turn.

A trajectory of T steps in this *factored reward* setting is τ=($\vec{a}^0$, $\vec{\mathbf{r}}^0$, $\vec{o}^1$, $\vec{a}^1$, $\vec{\mathbf{r}}^1$, $\ldots$, $\vec{o}^{T-1}$, $\vec{a}^{T-1}$, $\vec{\mathbf{r}}^{T-1}$). Here, $\vec{a}$, $\vec{\mathbf{r}}$ and $\vec{o}$ are vectors of all agents' actions, rewards and observations, respectively. An agent's reward in $\vec{\mathbf{r}}$, denoted by $\mathbf{r}_i$, is a *tuple* of the local costs and global rewards received by i based on the components in Eq. 11, and analogously for others. Since agents are cooperative, we focus on centralized learning, where our objective is to optimize a measure of joint rewards based on Eq. 11.

5.2. Joint Policy Iteration

The class of factored reward settings with agents that cooperate may be cast into the well-known framework of multiagent POMDPs (MPOMDPs) [7] — another special case of POSGs. MPOMDPs generalize POMDPs to multiple agents; actions and observations in an MPOMDP are a joint of the individual agent actions and observations. Importantly, the output is a *single* policy that maps joint observation sequences to joint actions. We define the MPOMDP below.

$$\text{MPOMDP} \triangleq \langle \mathcal{I}, S, A, \Omega, T, O, \mathcal{R} \rangle$$

- $\mathcal{I} = \{1, ..., Z\}$ is the set of Z interacting agents;
- S is the set of physical states;
- $A = A_1 \times ... \times A_Z$, is the set of joint actions where A_1, A_2, $\ldots$, A_Z are the sets of each agent's actions. A joint action is then, $\vec{a} = \{a_1, ..., a_Z\}$;
- $\Omega = \Omega_1 \times ... \times \Omega_Z$, is the set of joint observations where Ω_1, Ω_2, $\ldots$, Ω_Z are the sets of each agent's observations. A joint observation is $\vec{o} = \{o_1, ..., o_Z\}$;

Algorithm 3 MCES-MP

Require: Q-value table initialized to all 0s, and initial joint policy, π, that is greedy w.r.t Q-values; learning rate schedule α; error ϵ; and horizon T

1: Initialize count $c_{\vec{o},\vec{a}} \leftarrow 0$ for all $\vec{o}$ and $\vec{a}$
2: $m \leftarrow 0$
3: **repeat**
4: Pick joint observation history $\vec{o}$ and joint action $\vec{a}$
5: Set π to $\pi \leftarrow (\vec{o}, \vec{a})$
6: Generate trajectory τ of length T online by simulating transformed joint policy $\pi \leftarrow (\vec{o}, \vec{a})$
7: $Q_{\pi \leftarrow (\vec{o},\vec{a})} \leftarrow (1 - \alpha(m, c_{\vec{o},\vec{a}})) \cdot Q_{\pi \leftarrow (\vec{o},\vec{a})} + \alpha(m, c_{\vec{o},\vec{a}}) \cdot \mathcal{R}_{post-\vec{o}}(\tau)$
8: $c_{\vec{o},\vec{a}} \leftarrow c_{\vec{o},\vec{a}} + 1$
9: **if** $\max_{\vec{a}'} Q_{\pi \leftarrow (\vec{o},\vec{a}')} > Q_{\pi} + \epsilon(m, c_{\vec{o},\vec{a}}, c_{\vec{o},\pi(\vec{o})})$ **then**
10: $\pi(\vec{o}) \leftarrow \vec{a}'$ where $\vec{a}' \leftarrow \arg\max Q_{\pi \leftarrow (\vec{o},\vec{a}')}$
11: $m \leftarrow m + 1$
12: Reset $c_{\vec{o},\vec{a}} \leftarrow 0$ for all $\vec{o}$ and $\vec{a}$
13: **until** termination

- $T : S \times A \times S \rightarrow [0, 1]$ is the transition function that determines how the state evolves. It maps an origin state, a joint action, and an arrival state to a probability;
- $O : \Omega \times S \times A \rightarrow [0, 1]$ is the observation function that gives the informativeness of the observations toward the state. It maps an observation, the arrival state, and a joint action to a probability;
- $\mathcal{R} : S \times A \rightarrow \mathbb{R}$. While the factored-reward setting also includes the individual costs of each agent's actions, we obtain the single reward function that is needed as follows.

As a special case of POSGs with cooperative agents, an MPOMDP replaces the POSG's individual reward functions by a single *shared* reward function, $\mathcal{R}(s, \vec{a})$. In the factored reward setting introduced above, this joint reward function can be stated as:

$$\mathcal{R}(s, \vec{a}) = \sum\nolimits_{i \in \mathcal{I}} R_i(s, a_i) + R_G(s, \vec{a}) \tag{12}$$

Equation 12 characterizes the cumulative team reward, and represents the *potential function* of the cooperative game with individual heterogeneous rewards [26], where $R_i(s, a_i)$ is the individual payoff of agent i, and $R_G(s, \vec{a})$ is the positive externality from cooperating.

Algorithm 3, which we call MCES for MPOMDPs (MCES-MP), straightforwardly generalizes MCES-P. It modifies MCES-P in several ways. Line 4 picks joints instead of individual observations and actions. In line 5, $\pi \leftarrow (\vec{o}, \vec{a})$ denotes the *transformed policy* that prescribes $\vec{a}$ on encountering observation sequence $\vec{o}$. The trajectory τ in line 6 is as described in the previous subsection. Line 6 updates the Q-value by $R_{post-\vec{o}}$ with α as defined in Section 4.1. To define $\mathcal{R}_{post-\vec{o}}(\tau)$ in line 7, we begin by defining potential $\mathcal{R}(\tau)$ as $\mathcal{R}(\tau) = \sum_{t=0}^{T-1} \gamma^t \sum_{i \in \mathcal{I}} r_i^t + r_G^t$, where r_i^t and r_G^t are the local and global components of the reward $\mathbf{r}_i$ at time t received by i. $\mathcal{R}_{post-\vec{o}}(\tau)$ is then the portion of $\mathcal{R}(\tau)$ that succeeds joint observation $\vec{o}$.

We may instantiate the MCES-MP template using PALO bounds to obtain an algorithm MCES-MP-PALO that can be implemented similarly to the PALO instantiation of

MCES-P. A key difference from MCES-P is that the policy maps joints observations to joint actions, due to which the size of the local policy neighborhood N^{MP} is significantly larger. Specifically,

$$N^{\mathsf{MP}} = \prod_{i \in \mathcal{I}} |A_i| \left(\frac{\prod_{i \in \mathcal{I}} |\Omega_i|^T - 1}{\prod_{i \in \mathcal{I}} |\Omega_i| - 1} - 1 \right). \tag{13}$$

The other difference is in the maximal range of action-values of a policy π and its transformation π', denoted as $\Lambda(\pi, \pi')$ previously. We now utilize the maximum and minimum values of the reward function defined in Eq. 12 in the computation of Λ in Eq. 2. Given these changes, the definitions of k_m and $\epsilon(m, p, q)$ as in Eqs. 4 and 3 modify to accommodate them, and the algorithm terminates similarly. MCES-MP-PALO may terminate early similarly to MCES-P. Note again, that k_m is polynomial in problem parameters, particularly T and n, the problem horizon and number of agents, leading to *polynomial sample complexity per transform*.

Proposition 4 (Local optimality for MPOMDPs). *With probability $1 - \delta$, MCES-MP-PALO iterates over a series of policies mapping joint observations to joint actions, $\pi^1, \pi^2, \ldots, \pi^m$, such that the transformed policy π^{n+1} dominates the previous policy π^n in value for all agents and terminates to an ϵ-locally optimal policy π^m, where no neighbor dominates the converged policy by more than ϵ.*

The proof of this theorem follows from Greiner [24] in a straightforward way.

5.3. Joint Transformation of Individual Policies

Motivated by previous approaches in multiagent planning that divide the joint-policy search space into individual agent policy search spaces with a coordination mechanism [27, 28], our second method seeks to learn a vector of policies, one for each agent. A policy π_i for an agent i in this vector maps joint observations of all agents to the action prescribed for i. It does not require combining the rewards as in Eq. 12; rather it continues to utilize each agent's separate reward signal $\mathbf{r}_i$ obtained from $\mathcal{R}_i$. The model-free RL is outlined in Algorithm 4, and we refer to it as MCES for factored-reward MPOMDPs (MCES-FMP). Agents simultaneously act in a sequential environment with private observations that are conveyed exactly and perfectly to the centralized learner, and the centralized learner communicates the joint observations to each agent in the field. Agents perform their actions based on the joint observation sequence as prescribed by their respective policies. The entire trajectory of T time steps is then sent back to the learning algorithm.

Similar to MCES-MP, we begin by picking a joint observation history $\vec{o}$ and action $\vec{a}$ either randomly or in an iterated manner. However, it uses these to transform each agent's current policy by setting the action at $\vec{o}$ with its action in $\vec{a}$ (line 4). MCES-FMP maintains the Q-value for each agent's transformed policy (line 7) *additionally indexed by the joint of other agents' actions picked for $\vec{o}$.* This ensures consistent updates of Q-values for the same set of joint actions, and thus the multiagent policy vector. As such, it maintains as many Q-functions as the number of agents and combinations of other

agents' actions picked for $\vec{o}$ in the worst case, i.e., $\mathcal{O}(ZA^{Z-1})$. To obtain $\mathcal{R}_{i,post-\vec{o}}(\tau)$ in line 7 note that, $\mathcal{R}_i(\tau) = \sum_{t=0}^{T-1} \gamma^t (r_i^t + r_G^t)$. Then, $\mathcal{R}_{i,post-\vec{o}}(\tau)$ is simply the portion of $\mathcal{R}_i(\tau)$ that obtains after sequence $\vec{o}$ in the trajectory. Finally, the conjunction on line 9 ensures that all transformations are accepted together or none are.

Algorithm 4 MCES-FMP

Require: Q-value tables initialized to all 0s, and initial profile of agent policies, $\{\pi_i\}_{i=1}^Z$, that are greedy w.r.t. Q-values; learning rate schedule α; error ϵ; and horizon T

1: Initialize $c^i_{\vec{o},a_i} \leftarrow 0$ for all $\vec{o}$, a_i, and $i \in \mathcal{I}$, $m \leftarrow 0$
2: **repeat**
3: Pick joint observation history $\vec{o}$ and joint action $\vec{a}$
4: Set π_i to neighboring policy $\pi_i \leftarrow (\vec{o}, a_i)$ for all $i \in \mathcal{I}$
5: Generate trajectory τ of length T online by obtaining each agent's action using its transformed policy $\pi_i \leftarrow (\vec{o}, a_i)$ for each $i \in \mathcal{I}$
6: **for all** $i \in \mathcal{I}$ **do**
7: $Q^{\vec{a}_{-i}}_{\pi_i \leftarrow (\vec{o},a_i)} \leftarrow (1 - \alpha(m, c^i_{\vec{o},a_i})) \cdot Q^{a_{-i}}_{\pi_i \leftarrow (\vec{o},a_i)} + \alpha(m, c^i_{\vec{o},a_i}) \cdot \mathcal{R}_{i,post-\vec{o}}(\tau)$
8: $c^i_{\vec{o},a_i} \leftarrow c^i_{\vec{o},a_i} + 1$
9: **if** $\bigwedge_{i \in \mathcal{I}} \left(\max_{a'_i \in \mathcal{A}_i} Q^{\vec{a}_{-i}}_{\pi_i \leftarrow (\vec{o},a'_i)} > Q^{\vec{a}_{-i}}_{\pi_i} + \epsilon(n, c^i_{\vec{o},a_i}, c^i_{\vec{o},\pi_i(\vec{o})}) \right)$ **then**
10: $\pi_i(\vec{o}) \leftarrow a'_i$ where $a'_i \leftarrow \arg\max Q^{a_{-i}}_{\pi \leftarrow (\vec{o},a'_i)}$ $\forall i \in \mathcal{I}$
11: $m \leftarrow m + 1$
12: Reset $c^i_{\vec{o},a_i} \leftarrow 0$ for all $\vec{o}$, a_i, and $i \in \mathcal{I}$
13: **until** termination

An implication of line 9 is that if any agent receives a worse individual cost, despite the cumulative reward being better, MCES-FMP will not transform. While this may preclude higher team rewards in interim steps, the benefit is that MCES-FMP targets policies with higher joint values and takes larger steps in its search. Consequently, MCES-MP could approach the same local optima in policies as MCES-FMP, but may follow a different path as we demonstrate in Section 7.

When instantiated with PALO bounds (MCES-FMP-PALO), the policy decomposition approach provides significant sample complexity reductions. This is primarily because of a much reduced local neighborhood, where the first factor involving actions is not exponential in the number of agents (c.f. Eq. 13):

$$N^{\mathsf{FMP}} = |A_i| \left(\frac{\prod_{i \in \mathcal{I}} |\Omega_i|^T - 1}{\prod_{i \in \mathcal{I}} |\Omega_i| - 1} - 1 \right). \tag{14}$$

The conjunction on line 9 of MCES-FMP-PALO redefines Eqs. 3 and 4 as,

$$\epsilon^*(m, p) = \frac{\Lambda(\pi_i, \pi'_i)}{\sqrt{2p}} \sqrt{\ln \frac{\sqrt[2Z]{(4Z-2)(k_m - 1)} N^{\mathsf{FMP}}}{\sqrt[2Z]{\delta_m}}}. \tag{15}$$

$$k_m = \left\lceil 2 \left(\frac{\Lambda(\pi_i, \pi_i')}{\epsilon} \right)^2 \ln \left(\frac{\sqrt[2Z]{4Z-2} N^{\mathsf{FMP}}}{\sqrt[2Z]{\delta_m}} \right) \right\rceil \tag{16}$$

Here, the maximum range of action values $\Lambda(\pi_i, \pi_i')$ is computed analogously to Eq. 2 with the change that we utilize the maximum and minimum values of the reward function in Eq. 11. This range could get much narrower in comparison to the maximum range for MCES-MP-PALO because the rewards for the latter are a sum of all local costs. Of course, this benefits the sample bound k_m. Note again that k_m is a polynomial, leading to a polynomial sample complexity per transform.

Proposition 5. *The sample bound for MCES-FMP-PALO given in Eq. 16 is less than the sample bound for MCES-MP-PALO if the following holds for all agents:*

$$\ln \left(\frac{\sqrt[2Z]{4Z-2}\, N^{\mathsf{FMP}}}{\sqrt[2Z]{\delta_m}} \right) < \left(1 + \frac{\sum_{i \in \bar{\mathcal{I}}} \mathcal{R}_{i,max}}{\mathcal{R}_{i,max} + \mathcal{R}_{G,max}} \right)^2 \ln \frac{2N^{\mathsf{MP}}}{\delta_m} \tag{17}$$

where $\bar{\mathcal{I}}$ is the set of all agents other than agent i.

Proposition 5 is derived by simplifying the condition when the sample bound for MCES-FMP-PALO (Eq. 16) is smaller than that of MCES-MP-PALO (Eq. 4 with $\Lambda = \Lambda(\pi, \pi')$ and $N = N^{MP}$). MCES-FMP-PALO may terminate early if no neighboring policy exceeds the current value by $\epsilon - \epsilon^*(m, p)$. Its behavior is characterized by the following proposition.

Proposition 6 (Local optimality of factored policy). *With probability $1 - \delta$, MCES-FMP iterates over a joint set of individual policies mapping joint observations to individual actions $\pi^1, \pi^2, \ldots, \pi^m$ where every individual policy in the set $\pi^{n+1} = \langle \pi_1, \pi_2, \ldots, \pi_Z \rangle$ in the neighborhood of π^n dominates each agent's previous policy in value and terminates to an ϵ-locally optimal set of policies π^m. Here no neighbor for any agent dominates the converged policy by more than ϵ without another agent receiving a worse reward.*

In proving Proposition 6, the space of errors increases multiplicatively with the number of agents. In MCES-P, the agent may make two categories of errors: *transforming* or *terminating* erroneously due to noisy samples of the local neighborhood. In MCES-FMP, one or more agents may individually make these errors, growing the number of errors by $2Z$. The neighborhoods are factored into Z individual neighborhoods. These observations and the values in Eqs. 16 and 15 result in the bound δ. Exhaustive proofs of Propositions 5 and 6 are included in the supplementary material.

In summary, MCES-MP explores policies which map joint observations to joint actions, aggregating all local costs with the global reward. MCES-FMP explores the set of independent policies mapping joint observations to individual actions, where agents receive their own local cost with the global reward. The latter yields reduced sample complexity when instantiated with PALO guarantees.

6. Pruning Policy Search Space

Propositions 1, 3, 4, and 6 require exploring all local transformations of a policy for establishing local optimality. However, some of these transformations modify actions in response to observation sequences that are not likely to occur. Yet, we are required to obtain k_m samples of trajectories from real environments involving such observation sequences or establish a significant difference in action-values for fewer numbers of such samples. This contributes significantly to the empirical sample complexity of the algorithms as we noticed in our experiments. Subsequently, we seek ways to remove such rare observation sequences from consideration thereby pruning the policy search space. As these sequences are relatively much less likely they also contribute less to the expected value of a policy, but not considering them nonetheless introduces *regret* that we seek to compute.

Algorithm 5 MCES-IP_Prune

Require: Q-value table initialized to all 0s; initial policy, π_i, that is greedy w.r.t. Q-values; prior on set of models M_j; learning rate schedule α; horizon T; error ϵ; and regret bound ϕ

1: $c^{\vec{a}_j}_{\vec{o}_i,a_i} \leftarrow 0$, $c_{\vec{o}_i} \leftarrow 0$ for all $\vec{o}_i$, a_i, and a_j
2: $m \leftarrow 0$
3: $\mathcal{P} \leftarrow \emptyset$
4: **repeat**
5: Pick some observation history $\vec{o}_i$ and action a_i
6: $c_{\vec{o}_i} \leftarrow c_{\vec{o}_i} + 1$
7: **if** $\vec{o}_i \notin \mathcal{P}$ **then**
8: Modify π_i to $\pi_i \leftarrow (\vec{o}_i, a_i)$
9: **else**
10: Go to line 5
11: Generate trajectory τ of length T according to $\pi_i \leftarrow (\vec{o}_i, a_i)$
12: Generate belief sequence $\vec{b}_i$ based on τ using Eq. 5
13: Obtain most probable action sequence $\vec{a}_j$ from $\vec{b}_i$
14: $Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i,a_i} \leftarrow (1 - \alpha(m, c^{\vec{a}_j}_{\vec{o}_i,a_i})) \cdot Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i,a_i} + \alpha(m, c^{\vec{a}_j}_{\vec{o}_i,a_i}) \cdot R_{post-\vec{o}_i}(\tau)$
15: $c^{\vec{a}_j}_{\vec{o}_i,a_i} \leftarrow c^{\vec{a}_j}_{\vec{o}_i,a_i} + 1$
16: **if** $\max_{a'_i} Q^{\vec{a}_j}_{\pi_i \leftarrow \vec{o}_i,a'_i} \geq Q^{\vec{a}_j}_{\pi_i} + \epsilon^{\vec{a}_j}(m, c^{\vec{a}_j}_{\vec{o}_i,a_i}, c^{\vec{a}_j}_{\vec{o}_i,\pi_i(\vec{o}_i)})$ **then**
17: $\pi_i(\vec{o}_i) \leftarrow a'_i$
18: $m \leftarrow m + 1$
19: **for all** $\vec{o}_i, a_i, \vec{a}_j$ **do**
20: $c^{\vec{a}_j}_{\vec{o}_i,a_i} \leftarrow 0$
21: **if** $\sum_{\vec{o}_i \in \mathcal{P} \cup \vec{o}_i} \bar{regret}_{\vec{o}_i} \leq \phi + \rho(\sum_{\vec{o}_i} c_{\vec{o}_i})$ **then**
22: $\mathcal{P} \leftarrow \mathcal{P} \cup \vec{o}_i$
23: **until** termination

Ignoring a different action at some observation sequence $\vec{o}_i$ is regrettable because we are foregoing the possibility of improving the expected value of i's policy. Of course, a less likely observation sequence may not add much to the expected value of the current policy. Nevertheless, let ϕ be a user-defined bound on allowable regret. By avoiding transforming on $\vec{o}_i$, we are foregoing at most the largest post-$\vec{o}_i$ rewards; specifically

this regret is upper bounded by $\max_\tau R_{post-\vec{o}_i}(\tau) - \min_\tau R_{post-\vec{o}_i}(\tau)$. Regret on the expected value of the policy is bounded by

$$\begin{aligned} regret_{\vec{o}_i} &\leq Pr(\vec{o}_i; \pi_i, \pi_j) \left(\max_\tau R_{post-\vec{o}_i}(\tau) - \min_\tau R_{post-\vec{o}_i}(\tau) \right) \\ &= Pr(\vec{o}_i; \pi_i, \pi_j) \ (T - len(\vec{o}_i)) \ (R_{i,max} - R_{i,min}) \end{aligned} \tag{18}$$

Here, $Pr(\vec{o}_i; \pi_i, \pi_j)$ is the likelihood of the observation sequence $\vec{o}_i$ whose computation depends on the actions prescribed by both agents' policies, state transition and observation functions; rewards $R_{i,max}$ and $R_{i,min}$ are as defined previously. As the regret cannot exceed $T(R_{i,max} - R_{i,min})$, we may normalize the regret to obtain a proportion between 0 and 1 as, $\bar{regret}_{\vec{o}_i} = \frac{regret_{\vec{o}_i}}{T(R_{i,max} - R_{i,min})}$.

Of course, not knowing π_j and the model parameters implies that we cannot compute $Pr(\vec{o}_i; \pi_i, \pi_j)$ exactly. Hence, we settle for a crude approximation where $Pr(\vec{o}_i; \pi_i, \pi_j)$ is estimated by the fraction of times $\vec{o}_i$ occurs in the k_m or more samples generated so far. This requires keeping a count of each observation sequence encountered in the trajectories.

Let $\mathcal{P}$ be the set of i's observation sequences that will be avoided. If a bound ϕ on the normalized regret is given by the user, we may obtain $\mathcal{P}$ in a straightforward way: Sort the set of all observation sequences of all lengths by their frequency of occurrence in ascending order. Then, add a sequence into $\mathcal{P}$ beginning with the least frequent and moving up the ordering such that, $\sum_{\vec{o}_i \in \mathcal{P}} \bar{regret}_{\vec{o}_i} \leq \phi$. Consequently, a bound that is looser would allow disregarding more observation sequences to meet it and thereby prune a larger portion of the search space. Alternately, the sorting is not necessary and we may simply pick an observation sequence at random and check if adding it to $\mathcal{P}$ would cause the cumulative normalized regret to exceed ϕ.

The pruning mechanism described previously may be easily incorporated into MCES-P, -IP, -MP, and -FMP templates. Algorithm 5 outlines how it may be utilized with MCES-IP. Two additions can be observed. Lines 7-10 make a determination if the current $\vec{o}_i$ is in the set $\mathcal{P}$ of sequences that will not be considered (initialized to the empty set), and if not policy π_i is locally transformed at $\vec{o}_i$. Lines 21-22 add the observation sequence $\vec{o}_i$ into set $\mathcal{P}$ if the cumulative normalized regret due to all observation sequences in $\mathcal{P}$ including $\vec{o}_i$ remains less than or equal to given bound $\phi + \rho(\sum_{\vec{o}_i} c_{\vec{o}_i})$, where

$$\rho(c) = \begin{cases} 0 & c \geq k \\ +\infty & \text{otherwise} \end{cases} \tag{19}$$

Thus, $\mathcal{P}$ remains empty unless a reasonable number of samples are obtained to sufficiently approximate $Pr(\vec{o}_i; \pi_i, \pi_j)$. In the case of MCES-IP-PALO instantiation of the template, k could be simply set to the derived sample bound $\frac{k_m}{2}$. We ensure that sufficient samples containing $\vec{o}$, $c_{\vec{o}}$, are obtained before its regret is considered by defining $\rho(c_{\vec{o}})$ as 0 if $c_{\vec{o}} \geq \frac{k_m}{2}$ and $+\infty$ otherwise. Then, ρ is simply added to ϕ. The pruning mechanism shows a promising improvement as shown in Section 7. It improves the scalability of all methods by drastically reducing the search space.

7. Experiments

We implement MCES-IP-PALO, MCES-MP-PALO, and MCES-FMP-PALO to obtain converged policies for three self-interested multiagent domains and four cooperative multiagent domains. These domains range from being small to among the largest in the literature that have been utilized for evaluating multiagent RL. We discuss these domains below followed by analyzing the performance of the methods.

7.1. Problem Domain Specifications

Our first self-interested domain is the competitive, multiagent version of the Tiger problem [8]. Our second domain is a 3×2 autonomous unmanned aerial vehicle (AUAV) reconnaissance domain [29] shown in Fig. 1. Each agent has 3 actions: the AUAV may move left, right, or up, and the fugitive may move left, right, or down. Both agents receive 1 of 4 noisy public observations, representing whether both agents are East or West of each other, North or South of each other, in the same sector, or none of these. The subject agent, which is the AUAV, additionally receives 1 of 3 noisy private observations each correlated with an action the opponent takes. A third domain is the money laundering (ML) problem [30] where a blue team seeks to confiscate illicit money that the opponent red team is laundering. The red team can move money from the initial state to a series of placement states (banks and insurance), to layering states (offshore accounts and shell companies), to integration states (casinos and real estate), and to the safe clean pot. The blue team may place a sensor at each of these locations or confiscate the illicit funds. Each agent receives a noisy public observation indicating whether the money and sensor are in the same location, in the same laundering state, or neither. Blue team also receives a noisy observation of the last action of the red team.

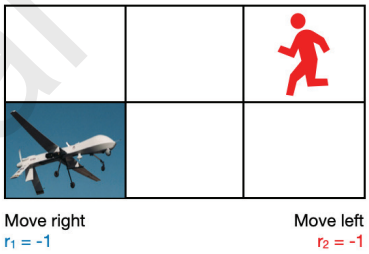
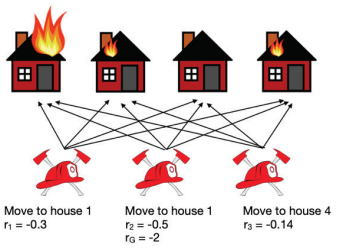


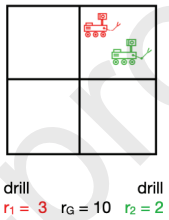
Figure 1: At the beginning of every round, the AUAV is positioned in the bottom-left sector and the fugitive is positioned in the top-right sector. All sectors in the leftmost column are considered safe.

For cooperative domains, our first domain is the well-known team Tiger problem with 2 agents [31]. We expand this domain to include factored rewards: the global reward corresponds to the original reward function, and additionally each agent $i \in \mathbb{N}$ incurs a cost of i for opening a door. The second domain is the 3-agent Firefighting (Fig. 2a) previously introduced for evaluating cooperative decentralized planning [17] using 4 houses with a maximum fire intensity of 3. Agents incur two local costs: the distance between houses if they move (distance$/(10-i)$) and the fire intensity of the

target ($f_h/(10-i)$). Our third domain is the 2- and 4-agent robot alignment problem (Fig. 3) [16] where each agent has two states and four states respectively. Robots incur local costs on turning ($-2-i$) and emitting IR ($-1-i$). Our final domain is the Mars Rovers (Fig. 2b) [32] where two agents conduct experiments at four possible sites by choosing to drill or sample. In order to receive the maximum reward, two of the sites require both agents to drill at the same time. Agents incur local costs on moving ($-1-i$), sampling at right sites ($2-i$), sampling at wrong sites ($1-i$), one agent drilling at right sites ($2-i$), both agent drilling at right sites ($4-i$), and drilling at wrong sites ($-10-i$).



(a)



(b)

Figure 2: (a) Illustration of an optimal policy on the firefighting domain. Agents incur two local costs: the distance between houses if they move (distance$/(10-i)$) and the fire intensity of the target ($f_h/(10-i)$). (b) The Mars Rover domain in which the top-right and bottom-left sites require both agents to drill at the same time. Agents receive ($-10-i$) for drilling at other sites. The agents receive ($2-i$) if sampling at top-left and bottom-right site respectively. Otherwise, they only receive ($1-i$).

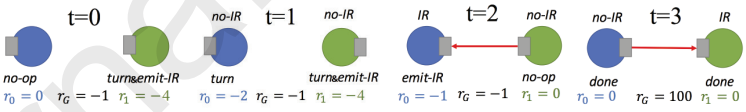


Figure 3: Illustration of a converged policy on the two-robot alignment problem yielding a trajectory from time step 0 to 3 where the robots align. Rewards r_0 and r_1 are local costs while r_G is the global team reward. Actions prescribed by the joint policy are mentioned below each robot.

Table 1 summarizes the statistics and parameter settings of all our domains. By setting δ to 0.1, all values are obtained at 99% confidence bounds. Convergence for larger domains is slightly more relaxed with $\epsilon = 0.1$. We set a generous time limit of 10 hours for MCES-P, MCES-IP, MCES-MP, and MCES-FMP on a standard PC (RHEL 6 Linux with 2.80 GHz Intel Xeon processor and 6 GB of main memory). This limit gives sufficient time to allow learning high-valued policies. Within this time, we sought to obtain the highest-valued converged policies from the all methods for smallest ϕ.

In the non-cooperative domains, each agent has a 15% chance of receiving a noisy public and private observation. The opponent in competitive games follows a single

Domain	Specifications	Opt. Value
Multiagent Tiger (Competitive)	$\epsilon = 0.05$, $\delta = 0.1$, $\phi = 0.15$, $T = 5, 6$, $\lvert S\rvert = 2$, $\lvert\Omega\rvert = 2$, $\lvert A_i\rvert = 3$, $\lvert A_j\rvert = 3$, $\lvert\Pi_j\rvert = 14$	4.35, 6.72
3×2 AUAV	$\epsilon = 0.1$, $\delta = 0.1$, $\phi = 0.2$, $T = 3$, $\lvert S\rvert = 36$, $\lvert\Omega\rvert = 4$, $\lvert A_i\rvert = 3$, $\lvert A_j\rvert = 3$, $\lvert\Pi_j\rvert = 4$	14.42
Money Laundering	$\epsilon = 0.1$, $\delta = 0.15$, $\phi = 0.2$, $T = 3$, $\lvert S\rvert = 99$, $\lvert\Omega\rvert = 4$, $\lvert A_i\rvert = 4$, $\lvert A_j\rvert = 5$, $\lvert\Pi_j\rvert = 8$	17.53
Multiagent Tiger (Cooperative)	$\epsilon = 0.1$, $\delta = 0.1$, $\phi = 0.1$, $T = 5, 6$, $\lvert\Omega\rvert = 2$, $\lvert A_i\rvert = 3$, $\lvert A_j\rvert = 3$	25.64, 32.11
Firefighting	$\epsilon = 0.1$, $\delta = 0.1$, $\phi = 0.15$, $T = 3, 4$, $\lvert S\rvert = 755$, $\lvert\Omega\rvert = 2$, $\lvert A_i\rvert = 4$, $\lvert A_j\rvert = 5$	-5.94, -4.83
Robot Alignment	$\epsilon = 0.1$, $\delta = 0.1$, $\phi = 0.15$, $T = 4$, $\lvert S\rvert = 4, 16$, $\lvert\Omega\rvert = 2$, $\lvert A_i\rvert = 4$, $\lvert A_j\rvert = 5$	14.39, 24.72
Mars Rovers	$\epsilon = 0.1$, $\delta = 0.1$, $\phi = 0.15$, $T = 4$, $\lvert S\rvert = 256$, $\lvert\Omega\rvert = 8$, $\lvert A_i\rvert = 6$, $\lvert A_j\rvert = 6$	20.19

Table 1: Parameter configurations with optimal policy values for the problem domains. The top three are competitive while the bottom four are cooperative. Notice that the domains range from having just two states (Tiger) to 755 (Firefighting).

policy (stationary environment) or fixed distribution over multiple policies (nonstationary environment). We simulate i's policies with opponent j following either a single policy or a mixture of two policies. These policies are picked from a predefined set Π_j. As per the policy space specified in Table 1, 105 games of the multiagent Tiger problem, 9 of the 3×2 AUAV problem, and 13 of ML comprise the data set.

7.2. Experiment Results

Monotonicity. First, we show that all four methods demonstrate the PALO guarantee of monotonically increasing in value across successive transformations. Figure 4 illustrates the average values from three runs for these methods with $\epsilon = 0$. Different runs undergo varying number of transformations with some policies not transforming at all because they are ϵ-locally optimal initially itself. As shown in Fig. 4, each successive transformation results in a higher value and in no case is the final policy lower in value than the initial policy, as we should expect. In every case, MCES-P-PALO requires more samples to transform than MCES-IP-PALO and MCES-MP-PALO requires more samples to transform than MCES-FMP-PALO.

Competitive baselines. Table 2 lists the mean of the sample bound k_m across the different stages m over all runs and the mean of the actual number of samples used, k_m^+, over all runs that were utilized by MCES-P-PALO and MCES-IP-PALO in competitive Tiger domain. k_m^+ is dominated by the samples that did not contribute because they lacked the observation sequence under consideration. We also compare MCES-P-PALO and MCES-IP-PALO with multiagent deep deterministic policy gradient (MADDPG) [19], which utilizes deep neural net representations, for completeness. MADDPG is a policy gradient based actor-critic method where the critic is assumed to have access to the other agent's policy as discussed in Section 2. However, to ensure a fair comparison, the MADDPG agent receives a private observation that noisily reveals information about the other agent's action to the critic. The numbers of samples used for training

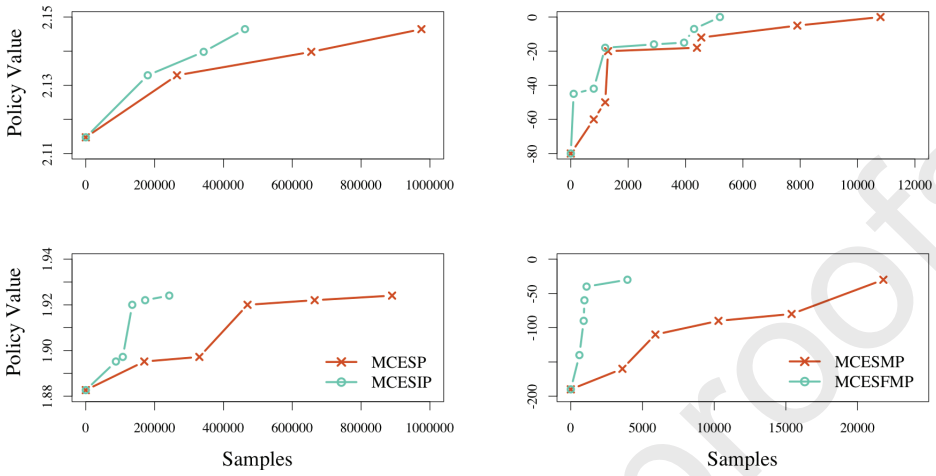


Figure 4: 'Samples' on the x-axis are the number of environmental interactions that the learner has experienced, and 'Policy Value' on the y-axis refers to the expected value of the policy learned with that number of samples. (**left**) Example policy transformation paths with intermediate values for single and mixed strategy opponents in the competitive Tiger. The bottom transformation paths are for an opponent with a mixed strategy. (**right**) Example policy transformation paths with intermediate values in the cooperative tiger (top) and robot alignment problem (bottom). Here, $\epsilon = 0$ for all methods.

		T	Initial deviation %	Final deviation %	Samples	Transforms	k_m^+
MCES-P	**Single**	5	56.09	19.20	83,663 ± 62	12 ± 1.4	89,430
	Mixed	5	56.09	25.24	102,002 ± 66	11 ± 1.4	113,901
	Single	6	43.95	16.01	234,752 ± 159	21 ± 1.2	249,029
	Mixed	6	43.95	21.84	292,778 ± 121	23 ± 1.4	314,920
MCES-IP	**Single**	5	56.09	**6.79**	29,351 ± 85	18 ± 1.2	78,932
	Mixed	5	56.09	**19.44**	34,640 ± 89	17 ± 1.4	79,171
	Single	6	43.95	**3.96**	73,927 ± 105	26 ± 1.4	150,342
	Mixed	6	43.95	**15.41**	104,357 ± 113	24 ± 1.2	159,556
MADDPG	**Single**	5	56.09	15.19	30,000	–	–
	Mixed	5	56.09	26.85	35,000	–	–
	Single	6	43.95	11.29	75,000	–	–
	Mixed	6	43.95	19.98	100,000	–	–

Table 2: Mean effective sample size and theoretical bound across the stages for the competitive Tiger problem, stratified over method and whether the opponent follows a single policy or a mixed set of policies. Deviations are shown as percentages relative to the known optimal policy values (lower is better). Pruning parameter ϕ is set to 0.1 for both methods. Initial policies are randomly generated, but competing methods use the same set of initial policies (hence the identical initial values). Lowest deviations for each setting among all methods are highlighted. ' – ' denotes not applicable.

		Initial deviation %	Final deviation %	Samples	Transforms	k_m^+
MCES-P	**Single**	70.25	17.96	32,397 $\pm$ 2,816	13 $\pm$ 0.4	91,443
	Mixed	70.25	23.79	42,126 $\pm$ 1,689	13 $\pm$ 1.1	96,328
MCES-IP	**Single**	70.25	**9.29**	6,437 $\pm$ 68	17 $\pm$ 1.3	20,397
	Mixed	70.25	**15.46**	19,499 $\pm$ 1,304	18 $\pm$ 0.7	22,763

Table 3: Mean effective sample size and theoretical bound across the stages for the competitive 3×2 AUAV problem, stratified over method and whether the opponent follows a single policy or a mixed set of policies. Deviations are shown as percentages relative to the known optimal policy values. Pruning parameter ϕ is set to 0.1 for both methods. Initial policies are randomly generated, but competing methods use the same set of initial policies (hence the identical initial values)

		Initial deviation %	Final deviation %	Samples	Transforms	k_m^+
MCES-P	**Single**	71.88	15.35	20,717 $\pm$ 2,418	8 $\pm$ 0.9	23,726
	Mixed	71.88	24.24	20,247 $\pm$ 4,974	9 $\pm$ 0.8	35,612
MCES-IP	**Single**	71.88	**7.64**	1,947 $\pm$ 330	12 $\pm$ 0.9	24,172
	Mixed	71.88	**14.32**	3,174 $\pm$ 536	13 $\pm$ 1.1	24,347

Table 4: Mean effective sample size and theoretical bound across the stages for the competitive Money Laundering problem, stratified over method and whether the opponent follows a single policy or a mixed set of policies. Deviations are shown as percentages relative to the known optimal policy values. Pruning parameter ϕ is set to 0.1 for both methods. Initial policies are randomly generated, but competing methods use the same set of initial policies (hence the identical initial values)

MADDPG are limited to match that used by MCES-IP-PALO during learning. Notice that the performance of this agent improves on MCES-P-PALO, as it considers the other agent's actions in its learning, but does not reach the quality of policies obtained by MCES-IP-PALO – it needs far more (about three times as many) samples to do so. Table 3 and Table 4 lists the mean of effective sample size and theoretical bound across the stages over all runs in 3×2 AUAV domain and Money Laundering domain. Pruning parameter ϕ is set to 0.1 for both methods. In validation of our theoretical result, MCES-IP-PALO requires remarkably fewer samples to transform per action sequence, about a quarter in Tiger and 3×2 AUAV, and nearly a tenth in ML. Additionally, the bound on k_m is also significantly less compared to the bound for MCES-P-PALO. Due to stochasticity in the simulations and finite sampling bounds, MCES-P-PALO and MCES-IP-PALO may deviate in transformation paths. However, in over 80% of the runs, both result in the same converged policy with MCES-IP-PALO converging on average under half the number of samples taken by MCES-P-PALO.

Cooperative baselines. We compare our method with a model-based RL technique for MPOMDPs, which offers the previous best solution. Bayes-adaptive factored-value POMCP (BA-FV-POMCP) [17] interleaves Bayesian model building with POMCP-

based planning to solve MPOMDPs. It uses the learned model obtained from the updated transition and observation count vectors at the beginning of an episode to build the simulator required by the POMCP rollouts. Another baseline we used for comparison is the direct cross-entropy policy search method [33] (DICEPS). It directly searches the space of joint policies. By adjusting some parameters, DICEPS allows us to directly control the number of sample used in each run. This feature is useful in comparing its performance with competing methods that use roughly the same number of samples. We also compare our methods with MADDPG on the tiger domain for completeness. We use approximately the same number of samples as MCES-MP-PALO and MCES-FMP-PALO to train MADDPG, and the critic receives a private observation that noisily reveals information about other agent's actions.

We empirically validate the drastically reduced number of samples required by MCES-FMP-PALO compared to MCES-MP-PALO. Additionally, the sample counts of the two methods are compared with those used by BA-FV-POMCP. As the simulations in BA-FV-POMCP are performed using the built model, we limit our attention to the number of samples utilized for model building. We report on the values of the converged joint policies for all four methods while allowing MCES-MP-PALO, BA-FV-POMCP, and DICEPS to use a similar number of samples as MCES-FMP-PALO. As a baseline, we implemented an algorithm that uses the single-agent MCES-P for learning the policy of each agent individually, but agents are still situated in the multiagent setting. Trajectories are generated by jointly performing the actions from each agent's policy, and each agent's individual reward is utilized for computing the action-value of the policy. To permit comparison, values of policies from all methods were computed similarly: the policies were simulated and local rewards of all agents were summed and added to the global reward; this was accumulated across all steps in the trajectory.

	T	Initial deviation %	Final deviation %	Samples (k_m)	Transforms	k_m^+	ϕ
MCES-FMP	5	908.11	**2.08**	32,594 $\pm$ 227	37 $\pm$ 1.1	79,482	0.1
	6	971.38	**1.83**	67,914 $\pm$ 104	59 $\pm$ 1.2	144,582	0.1
MCES-MP	5	908.11	8.23	33,985 $\pm$ 183	22 $\pm$ 1.4	80,484	0.1
	6	971.38	7.54	69,904 $\pm$ 177	36 $\pm$ 1.7	143,883	0.1
MCES-P	5	908.11	42.36	33,584 $\pm$ 82	29 $\pm$ 1.4	81,488	0.1
	6	971.38	14.14	68,391 $\pm$ 77	32 $\pm$ 1.6	144,505	0.1
BA FV-POMCP	5	908.11	34.63	32,993 $\pm$ 95	–	–	–
	6	971.38	8.44	67,884 $\pm$ 63	–	–	–
DICEPS	5	908.11	12.55	35,000	–	–	–
	6	971.38	9.76	70,000	–	–	–
MADDPG	5	908.11	33.04	35,000	–	–	–
	6	971.38	7.15	70,000	–	–	–

Table 5: Average metrics with std. error over 5 runs on the cooperative Tiger domain for all methods. Deviations are shown as percentages relative to the known optimal policy values (lower is better). Initial policies are randomly generated, but all competing methods use the same set of initial policies.

	A	Initial deviation %	Final deviation %	Samples (k_m)	Transforms	k_m^+	ϕ
MCES-FMP	2	1564.21	**1.33**	28,139 ± 31	16 ± 1.4	56,838	0.1
	4	1095.15	**1.71**	39,256 ± 180	36.8 ± 0.4	84,651	0.1
MCES-MP	2	1564.21	5.19	29,124 ± 32	10 ± 1.2	59,242	0.2
	4	1095.15	6.58	39,940 ± 101	33 ± 0.5	111,829	0.2
MCES-P	2	1564.21	20.34	26,564 ± 59	9 ± 0.3	70,962	0.3
	4	1095.15	22.42	40,303 ± 64	17 ± 1.1	99,035	0.3
BA FV-POMCP	2	1564.21	15.68	26,169 ± 18	–	–	–
	4	1095.15	16.29	40,040 ± 93	–	–	–
DICEPS	2	1564.21	4.22	35,000	–	–	–
	4	1095.15	5.03	40,000	–	–	–

Table 6: Average metrics with std. error over 5 runs on cooperative 2-agent and 4-agent Robot Alignment domain for all methods. Deviations are shown as percentages relative to the known optimal policy values. Initial policies are randomly generated, but all competing methods use the same set of initial policies.

	H.	Initial deviation %	Final deviation %	Samples (k_m)	Transforms	k_m^+	ϕ
MCES-FMP	3	23.23	**10.09**	20,921 ± 93	3.4 ± 0.5	40,580	0.1
	4	165.01	**9.17**	46,235 ± 95	40 ± 1.8	108,512	0.1
MCES-MP	3	76.77	58.33	21,940 ± 122	2.2 ± 0.4	53,904	0.2
	4	165.01	56.09	48,216 ± 155	26 ± 1.2	121,507	0.2
MCES-P	3	76.77	69.3	29,437 ± 56	1.7 ± 0.4	60,966	0.3
	4	165.01	67.38	45,991 ± 88	31 ± 1.3	120,883	0.3
BA FV-POMCP	3	76.77	60.53	20,962 ± 80	–	–	–
	4	165.01	54.83	46,270 ± 84	–	–	–
DICEPS	3	76.77	49.56	25,000	–	–	–
	4	165.01	34.01	50,000	–	–	–

Table 7: Average metrics with std. error over 5 runs on the cooperative Firefighting domain for all methods. Deviations are shown as percentages relative to the known optimal policy values. Initial policies are randomly generated, but all competing methods use the same set of initial policies.

Table 5 lists the metrics with the values averaged over 5 runs of MCES-MP-PALO, MCES-FMP-PALO, competing methods BA FV-POMCP, DICEPS, MADDPG, and baseline MCES-P-PALO for the cooperative Tiger domain. Tables 6, 7, and 8 lists these metrics for the multi-robot alignment, firefighting, and Mars rover domains, respectively. Note that the pruning parameter ϕ for MCES-MP-PALO and MCES-P-PALO are set to a higher value than MCES-FMP-PALO to make all three methods converge within the same amount of time. Each run starts at a differing initial policy that is generated randomly. However, the same set of initial policies is used for all methods to ensure a fair comparison. The reported k_m and k_m^+ are averaged over the 5 runs. The ϕ that led to convergence is also reported. Observe that MCES-FMP-PALO yields a joint policy that is significantly better than the joint from MCES-MP-PALO, BA FV-POMCP,

	Initial deviation %	Final deviation %	Samples (k_m)	Transforms	k_m^+	ϕ
MCES-FMP	112.13	**18.95**	46,099 $\pm$ 102	46 $\pm$ 2.4	12,910	0.1
MCES-MP	112.13	52.52	48,513 $\pm$ 128	37 $\pm$ 1.1	132,009	0.2
MCES-P	112.13	68.6	48,920 $\pm$ 79	28 $\pm$ 1.3	134,643	0.3
BA FV-POMCP	112.13	46.64	51,131 $\pm$ 20	–	–	–
DICEPS	112.13	51.50	50,000	–	–	–

Table 8: Average metrics with std. error over 5 runs on cooperative Mars Rover domain for all methods. Deviations are shown as percentages relative to the known optimal policy values. Initial policies are randomly generated, but all competing methods use the same set of initial policies.

DICEPS, and MADDPG for about the same numbers of contributing samples, k_m. This holds for multiple problem domains and their configurations. Furthermore, the policies generate coordination as is evident from the improvement over the MCES-P baseline. Model-based RL is generally believed to be more sample efficient than model-free methods, and Bayesian model-based RL essentially utilizes planning to decide between gathering samples that contribute to model-building versus exploitation. Nonetheless, MCES-FMP-PALO's comprehensive improvements over BA FV-POMCP is likely because policy iteration generally converges faster than value iteration [34] and reasonably accurate models for the evaluated domains (starting from a uniform prior over the models) may not be learned with the given number of samples. Along a similar vein, MADDPG needs many more samples than those provided to reach the policies obtained by MCES-MP-PALO despite being a policy search technique.

In Tables 2 – 8, the actual numbers of samples k_m^+ reported for our proposed methods may appear large for small problems, despite their theoretically polynomial sample complexity per transform. It should be noted that the reported sample counts appear large due to *rejection sampling*, where after selecting an $(\vec{o}, a)$, the learning agent keeps generating trajectories until it encounters a matching $\vec{o}$. For observation histories $\vec{o}$, many (especially the longer ones) may have a very low likelihood of being sampled, leading to many rejected samples. In fact, over 90% of the samples reported for MCES-IP, MCES-MP and MCES-FMP, are rejected, and not used by these algorithms. Though not used, we report these sample counts to give a complete picture and enable a fair comparison with competing methods.

Impact of pruning. Observation sequence pruning plays a crucial role in improving the scalability of all methods, dramatically reducing the search space and sample complexity while minimizing the impact on incurred regret. We list the mean of the total k_m^+ across all observation sequences for both problem domains in Table 9, both with and without pruning. As the policy search space for both MCES-MP and MCES-FMP is the same, the regret due to pruning the search space does not depend on the method used. Observation sequence pruning benefits both these methods equally in reducing the policy search space as we demonstrate in Table 9.

Each neighborhood is calculated with a horizon of 3. For example, for the multiagent Tiger problem, the size of the observation space per round is 6 (2 public $\times$ 3 private observations) with 3 possible actions, resulting in a maximum neighborhood of 128.

Method	Domain	Pruning	Neighborhood	k_m^+
MCES-IP	**Multiagent Tiger**	With	26	624,057
		Without	128	15,893,387
	AUAV	With	32	624,057
		Without	470	3,704,396
	ML	With	76	2,259,860
		Without	636	18,911,460
MCES-MP	**Multiagent Tiger**	With	46	2,540,664
		Without	324	17,784,648
	Firefighting	With	5,252	8,463,146
		Without	36,864	59,242,922
	Robot Alignment	With	5,252	1,694,019
		Without	36,864	11,858,138
MCES-FMP	**Multiagent Tiger**	With	3	4,057,404
		Without	18	28,401,834
	Firefighting	With	26	10,129,201
		Without	192	70,904,408
	Robot Alignment	With	26	1,997,067
		Without	192	13,979,472

Table 9: Neighborhood size and total k_m values for all the domains. Prunning parameter ϕ is set to 0.15 for the competitive Tiger, Firefighting, Robot Alignment, and Mars Rovers, 0.1 for cooperative Tiger, 0.2 for 3×2 AUAV and Money Laundering. Note that the total bound on samples reduces by almost an order of magnitude.

Given a regret bound of 0.15, 34 of 43 distinct observation sequences are eliminated on average, resulting in a neighborhood of 26.

Characterizing local optima. Figure 5 illustrates the values of converged policies and samples used for training with the regret ϕ varying from 0 to 0.2. Figure 5(left) shows a converged value of 2.47 by MCES-IP-PALO for the Tiger problem without pruning ($\phi = 0$) and Figure 5(right) shows a converged value of 10.28 for the policy vector in the absence of pruning in the team Tiger problem. This gradually drops as ϕ increases. Expectedly, the number of used samples drops significantly as well. We stop further increasing ϕ since the converged policies will be too far from optimal. With $\phi = 0$, both MCES-IP-PALO and MCES-FMP-PALO achieved near optimal policy values, albeit with more samples. Figure 3 demonstrated a successful run on the particularly challenging 2-robot Align as evidence of the good-quality policies learned by MCES-FMP. The learned policy vector guides the two robots to a successful alignment.

8. Concluding Remarks

MCES-P offers elegant policy-based RL in the partially observable, single-agent context. We generalized MCES-P to multiagent settings, introducing model-free learning by searching in the space of joint policies. The past decade's success in developing and scaling decision-theoretic planning for multiagent POMDPs has allowed attention to shift to realistic multiagent contexts. But, planning requires model specifications, which

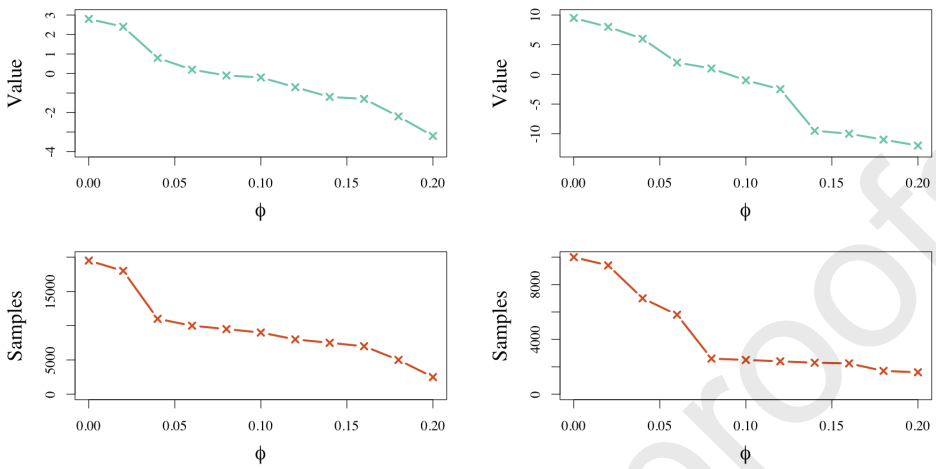


Figure 5: (**left**) Illustration of MCES-IP with varying ϕ, including convergence to near-optimal values with no pruning. Each data point averages 5 runs on Tiger T=3. (**right**) Illustration of MCES-FMP with varying ϕ, including convergence to near-optimal values with no pruning. Each data point averages 11 runs on team Tiger T=3. Standard errors are very small.

are hard to build for real-world problems with no prior information. Consequently, model-free reinforcement learning for these contexts has gained prominence.

We presented three templates for which PALO guarantees were established on the local optimality of the converged policies. While these guarantees do not speak to global optimality, nevertheless they offer a sound basis for the methods and guarantees of global optimality tend to be rare among RL techniques. Empirical results on three domains with differing numbers of agents comprehensively establish the competitive performances of these first model-free RL techniques for POSGs that fill an important gap in the literature on multiagent decision making.

In contrast to the original MCES-P template, our methods are more practicable for real-world multiagent reinforcement learning problems due to the reduced sample complexity. On the one hand, MCES-IP is suitable for competitive multiagent domains. It requires the presence of a private observation function for the learning agent and learns policies that map the agent's individual observations to its actions. With action prediction using beliefs over other agents' models, MCES-IP dramatically reduces the sample complexity of MCES-P-PALO with reductions on sample bounds and empirical sample counts ranging between 50% to 75% less than MCES-P-PALO. With observation sequence pruning, we further reduced over 80% samples required while introducing a normalized regret of between 0.15 and 0.2. MCES-IP is still able to achieve the same optima as MCES-P despite the fraction of samples required. On the other hand, MCES-MP and MCES-FMP focus on cooperative multiagent domains and do not require the presence of any model components. MCES-MP utilizes joint observations, actions and rewards instead of individual observations, actions, and rewards in MCES-P and MCES-IP. It explores policies which map joint observations to joint actions.

MCES-FMP differs from MCES-MP in that it still utilizes each agent's individual rewards instead of joint rewards. It explores the set of independent policies mapping joint observations to individual actions. All optima in MCES-MP are also optima in MCES-FMP, as a reduced total reward reflects a decrease in individual reward for some or all agents. However, a neighboring policy's total reward may be higher if the cost for one agent reduces more than that incurred by another agent. In this case, -FMP will not transform while -MP will. Therefore, optima in -FMP may not be so for -MP. Notwithstanding this, in practice, -FMP may route to this policy via a different path or arrive at similarly-valued optima. While both methods search through the same joint policy space, they differ in convergence criteria.

While the scalability of our method is fundamentally limited by the growth of N^{FMP} as the action and observation sets grow, it is noteworthy that the sample complexities of our methods are polynomial in the problem parameters, per transform. When the amount of useful interaction with the real environment is roughly equal, MCES-FMP learns significantly better quality policies than those by previous best methods, while being model-free and not requiring a simulator. Real-world interactions often dominate learning time (consider a mobile robot collecting samples using typically slow actuators) compared to interactions with a simulator. When judged from this holistic perspective, this article significantly advances the frontier of multiagent learning systems.

Finally, we ask: does the cooperative RL generalize to settings where the agents are structured into neighborhoods on a graph? Recall that both MCES-MP and MCES-FMP target the MPOMDP model, where the learning is centralized. In this setting, all agents pass their observations to the centralized learner. This is equivalent to agents being aware of all other agents' observations. Segmenting the cooperative agents into neighborhoods limits information exchange between the agents, and MCES-MP and MCES-FMP may not be applied where information exchange is limited among agents. MCES-IP, however, is precisely meant for such general settings where agents' observation exchanges are limited.

One line of further inquiry would be to mitigate the inflated actual sample complexity due to rejection sampling. While pruning addresses this limitation to some extent, other methods need to be investigated for deeper impact. It is noteworthy that the reason our methods cannot avoid rejection sampling is that we only allow realistic interactions between our learners and the environment, as opposed to having a greater control on sampling using a simulator. If the learner had access to a simulator, then importance sampling could be used [35] to significantly reduce the sample complexity in practice. As another next step, we are studying recent advances in using deep recurrent RL for POMDPs [36], and the challenges in generalizing the network and samples to the exponentially-harder POSGs.

9. References

[1] R. Powers, Y. Shoham, T. Vu, A general criterion and an algorithmic framework for learning in multi-agent systems, Machine Learning 67 (1) (2007) 45–76.

[2] R. S. Sutton, A. G. Barto, Introduction to reinforcement learning, MIT Press, 1998.

[3] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) 279–292.

[4] T. J. Perkins, Reinforcement learning for pomdps based on action values and stochastic optimization, in: Association for the Advancement of Artificial Intelligence, 2002, pp. 199–204.

[5] L. Peshkin, K.-E. Kim, N. Meuleau, L. Kaelbling, Learning to cooperate via policy search, in: Uncertainty in Artificial Intelligence, 2000, pp. 489–496.

[6] D. S. Bernstein, S. Zilberstein, N. Immerman, The complexity of decentralized control of markov decision processes, in: Uncertainty in Artificial Intelligence, 2000, pp. 32–37.

[7] J. V. Messias, M. Spaan, P. U. Lima, Efficient offline communication policies for factored multiagent POMDPs, in: Neural Information Processing Systems, 2011, pp. 1917–1925.

[8] P. J. Gmytrasiewicz, P. Doshi, A framework for sequential planning in multiagent settings, Journal of Artificial Intelligence Research 24 (2005) 49–79.

[9] R. Ceren, P. Doshi, B. Banerjee, Reinforcement learning in partially observable multiagent settings: Monte carlo exploring policies with pac bounds, in: International Conference On Autonomous Agents and Multi-Agent Systems, 2016, pp. 530–538.

[10] L. P. Kaelbling, M. Littman, A. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence.

[11] P. Doshi, Decision making in complex multiagent contexts: A tale of two frameworks, AI Magazing 4 (2012) 82.

[12] B. Ng, K. Boakye, C. Meyers, A. Wang, Bayes-adaptive interactive pomdps, in: Association for the Advancement of Artificial Intelligence, 2012.

[13] T. N. Hoang, K. H. Low, A general framework for interacting bayes-optimally with self-interested agents using arbitrary parametric model and model prior, in: International Joint Conferences on Artificial Intelligence, 2013, pp. 1394–1400.

[14] B. Banerjee, J. Lyle, L. Kraemer, R. Yellamraju, Sample bounded distributed reinforcement learning for decentralized pomdps., in: Association for the Advancement of Artificial Intelligence, 2012, pp. 1256–1262.

[15] L. Kraemer, B. Banerjee, Reinforcement learning of informed initial policies for decentralized planning, Transactions on Autonomous and Adaptive Systems 9 (4) (2014) 18.

[16] L. Kraemer, B. Banerjee, Multi-agent reinforcement learning as a rehearsal for decentralized planning, Neurocomputing 190 (2016) 82–94.

[17] C. Amato, F. A. Oliehoek, Scalable planning and learning for multiagent pomdps, in: Association for the Advancement of Artificial Intelligence, 2015, pp. 1995–2002.

[18] S. Omidshafiei, J. Pazis, C. Amato, J. How, V. John, Deep decentralized multi-task multi-agent rl under partial observability, in: International Conference on Machine Learning, 2017.

[19] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Neural Information Processing Systems, 2017, pp. 6382–6393.

[20] J. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, I. Mordatch, Learning with opponent-learning awareness, in: International Conference on Autonomous Agents and MultiAgent Systems, 2018, pp. 122–130.

[21] Z. Cao, C.-T. Lin, Reinforcement Learning from Hierarchical Critics, arXiv e-prints (2019) arXiv:1902.03079arXiv:1902.03079.

[22] J. Gupta, M. Egorov, M. Kochenderfer, Cooperative multi-agent control using deep reinforcement learning, in: International Conference On Autonomous Agents and Multi-Agent Systems, 2017, pp. 66–83.

[23] M. Liu, X. Liao, L. Carin, The infinite regionalized policy representation, in: International Conference on Machine Learning, 2011, pp. 769–776.

[24] R. Greiner, Palo: A probabilistic hill-climbing algorithm, Artificial Intelligence 84 (1) (1996) 177–208.

[25] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations, Cambridge University Press, 2009.

[26] D. Monderer, L. Shapley, Potential games, Games and Economic Behavior.

[27] C. Amato, D. S. Bernstein, S. Zilberstein, Optimizing memory-bounded controllers for decentralized pomdps, in: International Joint Conferences on Artificial Intelligence, 2007, pp. 1–8.

[28] C. Guestrin, D. Koller, R. Parr, Multiagent planning with factored mdps., in: Neural Information Processing Systems, Vol. 1, 2001, pp. 1523–1530.

[29] E. Sonu, P. Doshi, Generalized and bounded policy iteration for finitely-nested interactive pomdps: Scaling up, in: International Conference On Autonomous Agents and Multi-Agent Systems, 2012, pp. 1039–1048.

[30] B. Ng, C. Meyers, K. Boakye, J. Nitao, Towards applying interactive POMDPs to real-world adversary modeling, in: Innovative Applications of Artificial Intelligence, 2010, pp. 1814–1820.

[31] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, S. Marsella, Taming decentralized pomdps: Towards efficient policy computation for multiagent settings, in: International Joint Conferences on Artificial Intelligence, 2003, pp. 705–711.

[32] C. Amato, S. Zilberstein, Achieving goals in decentralized pomdps, in: International Conference On Autonomous Agents and Multi-Agent Systems, 2009, pp. 593–600.

[33] F. Oliehoek, J. Kooij, N. Vlassis, The cross-entropy method for policy search in decentralized pomdps, Informatica 32 (2008) 341–357.

[34] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall, Egnlewood Cliffs, 2010.

[35] K. A. Ciosek, S. Whiteson, OFFER: Off-Environment Reinforcement Learning, in: Association for the Advancement of Artificial Intelligence, 2017, pp. 1819–1825.

[36] M. J. Hausknecht, P. Stone, Deep recurrent q-learning for partially observable mdps, CoRR abs/1507.06527. arXiv:1507.06527.
URL http://arxiv.org/abs/1507.06527

APPENDIX

In this section, we provide detailed proofs of the propositions introduced in the article.

Proof of Proposition 1

Proof. MCES-P-PALO in the multiagent setting allows for a PALO-style guarantee of ϵ-local optimality. To show that the total error for MCES-P-PALO is bounded by the user-defined δ for a given ϵ, we must first define the types of errors that can occur in selecting dominating neighboring policies and terminating when none is found after sampling.

a After seeing n samples (where $n < k$), MCES-P-PALO selects some $\pi' \in neighbor(\pi)$, as π' appears better than π, but, in fact, it is not better

b After seeing n samples (where $n < k$), MCES-P-PALO cannot find a $\pi' \in neighbor(\pi)$ where π' is better than π, but, in fact, there is one that is much better

c After seeing all k samples, MCES-P-PALO selects some $\pi' \in neighbor(\pi)$, as π' appears better than π, but, in fact, it is not better

d After seeing all k samples, MCES-P-PALO cannot find a $\pi' \in neighbor(\pi)$ where π' is better than π, but, in fact, there is one that is much better

Recall ϵ and k_m for MCES-P in the multiagent setting are as follows.

$$\epsilon^*(m,i,k_m) = \frac{\Lambda(\pi_i^m,\pi_i')}{\sqrt{2i}}\sqrt{\ln\frac{2(k_m-1)\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}{|\Pi_j|\delta_m}}$$

$$k_m = \left\lceil 2\left(\frac{\Lambda(\pi_i^m,\pi_i')}{\epsilon}\right)^2 \ln\frac{2\sum_{\pi_j}|\mathcal{N}(\pi_i^m|}{|\Pi_j|\delta_m}\right\rceil$$

First, we show the theoretical proof for MCES-P-PALO in the multiagent setting. That is, considering an additional agent is interacting with the environment, MCES-P-PALO holds with the following modifications.

$$\begin{aligned}
a^n_{m,\pi_j} =& Pr[\exists\pi_i' \in \mathcal{N}(\pi_i^m) : (Q^{\pi_i'} - Q^{\pi_i^m}) \geq \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m}) \text{ and } R_i(\pi_i',\pi_j) < R_i(\pi_i^m,\pi_j)]\\
b^n_{m,\pi_j} =& Pr[\exists\pi_i' \in \mathcal{N}(\pi_i^m) : (Q^{\pi_i'} - Q^{\pi_i^m}) < \epsilon - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m}) \text{ and } R_i(\pi_i',\pi_j) > R_i(\pi_i^m,\pi_j) + \epsilon]\\
c_{m,\pi_j} =& Pr[\exists\pi_i' \in \mathcal{N}(\pi_i^m) : (Q^{\pi_i'} - Q^{\pi_i^m}) \geq \frac{\epsilon}{2} \text{ and } R_i(\pi_i',\pi_j) < R_i(\pi_i^m,\pi_j)]\\
d_{m,\pi_j} =& Pr[\exists\pi_i' \in \mathcal{N}(\pi_i^m) : (Q^{\pi_i'} - Q^{\pi_i^m}) < \frac{\epsilon}{2} \text{ and } R_i(\pi_i',\pi_j) > R_i(\pi_i^m,\pi_j) + \epsilon]
\end{aligned}$$

We can represent each of the $Pr[\cdot]$ above as disjoint sets over neighbors and the other agents' policies. As an example, $\mathcal{N}_i^<(\pi_i^m) = \{\pi_i \in \mathcal{N}(\pi^m) | R_i(\pi_i' < R_i(\pi_i^m,\pi_j)\}$. Additionally, let $R_i(\pi_i,\pi_j)$ be the true reward for agent i following policy π_i when agent j follows π_j.

$$\begin{aligned}
a^n_{m,\pi_j} &= Pr[\bigvee_{\pi_i'\in\mathcal{N}_i^<(\pi_i^m)} (Q^{\pi_i'} - Q^{\pi_i^m}) \geq \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m})]\\
b^n_{m,\pi_j} &= Pr[\bigvee_{\pi_i'\in\mathcal{N}_i^>(\pi_i^m)} (Q^{\pi_i'} - Q^{\pi_i^m}) < \epsilon - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m})]\\
c_{m,\pi_j} &= Pr[\bigvee_{\pi_i'\in\mathcal{N}_i^<(\pi_i^m)} (Q^{\pi_i'} - Q^{\pi_i^m}) \geq \frac{\epsilon}{2}] \qquad d_{m,\pi_j} = Pr[\bigvee_{\pi_i'\in\mathcal{N}_i^>(\pi_i^m} (Q^{\pi_i'} - Q^{\pi_i^m}) < \frac{\epsilon}{2}]
\end{aligned}$$

Considering every possible π' in π^m's neighborhood, assuming a fixed π_j leads to the summation:

$$\begin{aligned}
a^n_{m,\pi_j} \leq& \sum_{\pi_i'\in\mathcal{N}_i^<(\pi_i^m)} Pr[(Q^{\pi_i'} - Q^{\pi_i^m}) \geq \epsilon(m, c_i^{\pi_i'}, c_i^{\pi^m}, k_m)]\\
&\leq \sum_{\pi_i'\in\mathcal{N}_i^<(\pi_i^m)} Pr[(Q^{\pi_i'} - Q^{\pi_i^m}) \geq (R_i(\pi_i',\pi_j) - R_i(\pi_i^m,\pi_j)) + \epsilon(m, c_i^{\pi_i'}, c_i^{\pi^m})]\\
&\leq \sum_{\pi_i'\in\mathcal{N}_i^<(\pi_i^m)} \exp\{-2i\left(\frac{\epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m})}{\Lambda(\pi_i',\pi_i^m,\pi_j)}\right)^2\} \qquad (20)\\
&= |\mathcal{N}_i^<(\pi_i^m)|\frac{\delta_m|\Pi_j|}{2(k_m-1)\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}
\end{aligned}$$

We reduce 20 by using ϵ when $i = j < k_m$. When we consider every possible π_j, we get the total sum probability of error for type a.

$$a_m^n = \sum_{\pi_j} a_{m,\pi_j}^n Pr(\pi_j) \leq \frac{1}{|\Pi_j| \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \cdot \sum_{\pi_j} |\mathcal{N}_i^<(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2(k_m - 1)}$$

We assume here that $Pr(\pi_j)$ is the uniform distribution, and, therefore, is $\frac{1}{|\Pi_j|}$. b_m^n is expressed similarly.

$$\begin{aligned} b_{m,\pi_j}^n &\leq \sum_{\pi_i' \in \mathcal{N}_i^>(\pi_i^m)} Pr\left[(Q^{\pi_i'} - Q^{\pi_i^m}) < \epsilon - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m}, k_m)\right] \\ &\leq \sum_{\pi_i' \in \mathcal{N}_i^>(\pi_i^m)} Pr[(Q^{\pi_i'} - Q^{\pi_i^m}) \leq (R_i(\pi_i', \pi_j) - R_i(\pi_i^m, \pi_j)) - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m}, k_m)] \\ &\leq \sum_{\pi_i' \in \mathcal{N}_i^>(\pi_i^m)} \exp\{-2i \left(\frac{-\epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m})}{\Lambda(\pi_i', \pi_i^m, \pi_j)}\right)^2\} \quad \leq \quad |\mathcal{N}_i^>(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2(k_m - 1) \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \end{aligned}$$

As with a_m^n, we consider every possible π_j.

$$b_m^n = \sum_{\pi_j} b_{m,\pi_j}^n Pr(\pi_j) \leq \frac{1}{|\Pi_j| \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \sum_{\pi_j} |\mathcal{N}_i^>(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2(k_m - 1)}$$

The other two error types, where $n = k_m$, follow similarly, but substitute the current k_m for n.

$$c_{m,\pi_j} \leq \sum_{\pi_i' \in \mathcal{N}_i^<(\pi_i^m)} Pr\left[(Q^{\pi_i'} - Q^{\pi_i^m}) \geq \frac{\epsilon}{2}\right] \quad \leq \quad \sum_{\pi_i' \in \mathcal{N}_i^<(\pi_i^m)} \exp\{-2i \left(\frac{\epsilon/2}{\Lambda(\pi_i', \pi_i^m, \pi_j)}\right)^2\} \tag{21}$$

$$= |\mathcal{N}_i^<(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2 \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \tag{22}$$

Where 22 is obtained by substituting k_m with its derived value in 21 and reducing.

$$\begin{aligned} c_m &= \sum_{\pi_j} c_{m,\pi_j}^n Pr(\pi_j) \leq \frac{1}{|\Pi_j| \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \sum_{\pi_j} |\mathcal{N}_i^<(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2} \\ d_{m,\pi_j} &\leq \sum_{\pi_i' \in \mathcal{N}_i^>(\pi_i^m)} Pr\left[(Q^{\pi_i'} - Q^{\pi_i^m}) < \frac{\epsilon}{2}\right] \\ &\leq \sum_{\pi_i' \in \mathcal{N}_i^>(\pi_i^m)} \exp\{-2i \left(\frac{\epsilon/2}{\Lambda(\pi_i', \pi_i^m, \pi_j)}\right)^2\} \quad \leq |\mathcal{N}_i^>(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2 \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \\ d_m &= \sum_{\pi_j} d_{m,\pi_j}^n Pr(\pi_j) \leq \frac{1}{|\Pi_j| \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \sum_{\pi_j} |\mathcal{N}_i^>(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2} \end{aligned}$$

The sum total of our error is as follows.

$$
\begin{aligned}
&\sum_{n=1}^{k_m-1}[a_m^n + b_m^n] + c_m + d_m \\
&\quad= \sum_{n=1}^{k_m-1}\left[\frac{1}{|\Pi_j|\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}\sum_{\pi_j}|\mathcal{N}_i^<(\pi_i^m)|\frac{\delta_m|\Pi_j|}{2(k_m-1)} + \frac{1}{|\Pi_j|\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}\sum_{\pi_j}|\mathcal{N}_i^>(\pi_i^m)|\frac{\delta_m|\Pi_j|}{2(k_m-1)}\right] \\
&\qquad + \frac{1}{|\Pi_j|\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}\sum_{\pi_j}|\mathcal{N}_i^<(\pi_i^m)|\frac{\delta_m|\Pi_j|}{2} + \frac{1}{|\Pi_j|\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}\sum_{\pi_j}|\mathcal{N}_i^>(\pi_i^m)|\frac{\delta_m|\Pi_j|}{2} \\
&\quad= \frac{\delta_m|\Pi_j|}{2|\Pi_j|\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}\left(\sum_{n=1}^{k_m-1}\frac{1}{k_m-1}\left[\sum_{\pi_j}|\mathcal{N}_i^<(\pi_i^m)| + \sum_{\pi_j}|\mathcal{N}_i^>(\pi_i^m)|\right] + \sum_{\pi_j}|\mathcal{N}_i^<(\pi_i^m)| + \sum_{\pi_j}|\mathcal{N}_i^>(\pi_i^m)|\right) \\
&\quad= \frac{\delta_m}{2\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}\left(\sum_{n=1}^{k_m-1}\frac{1}{k_m-1}\sum_{\pi_j}(|\mathcal{N}_i^<(\pi_i^m)| + |\mathcal{N}_i^>(\pi_i^m)|) + \sum_{\pi_j}(|\mathcal{N}_i^<(\pi_i^m)| + |\mathcal{N}_i^>(\pi_i^m)|)\right) \\
&\quad\leq \frac{\delta_m}{2\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}\left(\sum_{n=1}^{k_m-1}\frac{1}{k_m-1}\sum_{\pi_j}|\mathcal{N}(\pi_i^m)| + \sum_{\pi_j}|\mathcal{N}(\pi_i^m)|\right) \\
&\quad= \frac{\delta_m}{2\sum_{\pi_j}|\mathcal{N}(\pi_i^m)|}(k_m-1)\frac{1}{k_m-1}\cdot 2\sum_{\pi_j}|\mathcal{N}(\pi_i^m)| = \delta_m
\end{aligned}
$$

It then follows that, over all possible transformations, the total error is bounded by δ.

$$\sum_{m=1}^{\infty}\delta_m = \sum_{m=1}^{\infty}\frac{6\delta}{m^2\pi^2} = \frac{6\delta}{\pi^2}\sum_{m=1}^{\infty}\frac{1}{m^2} = \frac{6\delta}{\pi^2}\frac{\pi^2}{6} = \delta$$

□

Proof of Proposition 2

Proof. Consider the following expanded definition of the above proposition.

$$
\begin{aligned}
&\sum_{t=0}^{T}[\max_s\{R(a_{\pi_i'}^t, a_j^t, s) - R(a_{\pi_i^k}^t, a_j^t, s)\} - \min_s\{R(a_{\pi_i'}^t, a_j^t, s) - R(a_{\pi_i^k}^t, a_j^t, s)\}] \leq \\
&\sum_{t=0}^{T}[\max_{s,a_j}\{R(a_{\pi_i'}^t, a_j, s) - R(a_{\pi_i^k}^t, a_j, s)\} - \min_{s,a_j}\{R(a_{\pi_i'}^t, a_j, s) - R(a_{\pi_i^k}^t, a_j, s)\}]
\end{aligned}
$$

(1) Assume $\min_s = \min_{s,a_j}$ for all $t \in T$, resulting in the expression $\sum_{t=0}^{T}\max_s\{R(a_{\pi_i'}^t, a_j^t, s) - R(a_{\pi_i^k}^t, a_j^t, s)\} \leq \sum_{t=0}^{T}\max_{s,a_j}\{R(a_{\pi_i'}^t, a_j, s) - R(a_{\pi_i^k}^t, a_j, s)\}$. It must be the case that the LHS must be at most equivalent to the RHS, as, if the a_j selected in the LHS is the maximal value, the $\max_{s,a_j}$ will select it. If the a_j selected on the LHS is not the maximal value for a given a_i and s, the RHS must then be greater.

(2) Assume $\max_s = \max_{s,a_j}$ for all $t \in T$, resulting in the expression $\sum_{t=0}^{T} \min_s\{R(a^t_{\pi'_i}, a^t_j, s) - R(a^t_{\pi^k_i}, a^t_j, s)\} \geq \sum_{t=0}^{T}[\min_{s,a_j}\{R(a^t_{\pi'_i}, a_j, s) - R(a^t_{\pi^k_i}, a_j, s)\}]$. Analogously to point 1, if the a_j on the LHS is the minimal value, then the $\min_{s,a_j}$ on the RHS must select it. If it is not, the RHS must be less.

Following point 1 and 2, since each component of the LHS must be bounded by the equivalent component on the RHS, the LHS must be a smaller range than the RHS. Therefore, $\Lambda^{\vec{a_j}}(\pi'_i, \pi^m_i) \leq \Lambda(\pi'_i, \pi^m_i, \pi_j)$

□

Proof of Proposition 3

Proof. MCES-IP follows similarly from MCES-P-PALO in the multiagent setting, but additionally predicates rewards on the belief of the other opponent. Since we include a dimension over the belief vector, ϵ and k_m are augmented in the following fashion.

$$\epsilon * (m, i, k_m) = \frac{\Lambda^{\vec{a_j}}(\pi^m_i, \pi'_i)}{\sqrt{2i}} \sqrt{\ln \frac{2(k_m - 1)\sum_{\pi_j} |\mathcal{N}(\pi^m_i)|}{|\Omega_i|^{-T} |\Pi_j| \delta_m}}$$

$$k_m = \left\lceil 2 \left(\frac{\Lambda(\pi^m_i, \pi'_i)}{\epsilon} \right)^2 \ln \frac{2|\Omega_i|^T \sum_{\pi_j} |\mathcal{N}(\pi^m_i)|}{|\Pi_j| \delta_m} \right\rceil$$

The probability for error is categorized in the same way as MCES-P-PALO, with the aforementioned change in the context of beliefs over opponent behavior included.

$$\begin{aligned}
a^n_{m,\pi_j} &= Pr[\exists \vec{b_i} \exists \pi'_i \in \mathcal{N}(\pi^m_i) : (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) \geq \epsilon(m, c_i^{\pi'_i, \vec{b_i}}, c_i^{\pi^m_i, \vec{b_i}}, k_m) \\
&\qquad \text{and } R_i(\pi'_i, \pi_j) < R_i(\pi^m_i, \pi_j)] \\
b^n_{m,\pi_j} &= Pr[\exists \vec{b_i} \exists \pi'_i \in \mathcal{N}(\pi^m_i) : (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) < \epsilon - \epsilon(m, c_i^{\pi'_i, \vec{b_i}}, c_i^{\pi^m_i, \vec{b_i}}, k_m) \\
&\qquad \text{and } R_i(\pi'_i, \pi_j) > R_i(\pi^m_i, \pi_j) + \epsilon] \\
c_{m,\pi_j} &= Pr[\exists \vec{b_i} \exists \pi'_i \in \mathcal{N}(\pi^m_i) : (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) \geq \frac{\epsilon}{2} \text{ and } R_i(\pi'_i, \pi_j) < R_i(\pi^m_i, \pi_j)] \\
d_{m,\pi_j} &= Pr[\exists \vec{b_i} \exists \pi'_i \in \mathcal{N}(\pi^m_i) : (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) < \frac{\epsilon}{2} \text{ and } R_i(\pi'_i, \pi_j) > R_i(\pi^m_i, \pi_j) + \epsilon]
\end{aligned}$$

Similarly to MCES-P-PALO, we can convert this to a disjoint set of neighbors.

$$\begin{aligned}
a^n_{m,\pi_j} &= Pr[\bigvee_{\vec{b_i}} \bigvee_{\pi'_i \in \mathcal{N}_i^<(\pi^m_i)} (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) \geq \epsilon(m, c_i^{\pi'_i, \vec{b_i}}, c_i^{\pi^m_i, \vec{b_i}})] \\
b^n_{m,\pi_j} &= Pr[\bigvee_{\vec{b_i}} \bigvee_{\pi'_i \in \mathcal{N}_i^>(\pi^m_i)} (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) < \epsilon - \epsilon(m, c_i^{\pi'_i, \vec{b_i}}, c_i^{\pi^m_i, \vec{b_i}})] \\
c_{m,\pi_j} &= Pr[\bigvee_{\vec{b_i}} \bigvee_{\pi'_i \in \mathcal{N}_i^<(\pi^m_i)} (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) \geq \frac{\epsilon}{2}] \quad d_{m,\pi_j} = Pr[\bigvee_{\vec{b_i}} \bigvee_{\pi'_i \in \mathcal{N}_i^>(\pi^m_i)} (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi^m_i}_{\vec{b_i}}) < \frac{\epsilon}{2}]
\end{aligned}$$

Here we will reintroduce Λ instead of the range present in MCES-P-PALO.

$$a^n_{m,\pi_j} = Pr[\bigvee_{\vec{b_i}} \bigvee_{\pi'_i \in \mathcal{N}_i^<(\pi_i^m)} (Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi_i^m}_{\vec{b_i}}) \geq \epsilon(m, c_i^{\pi'_i,\vec{b_i}}, c_i^{\pi_i^m,\vec{b_i}}, k_m)]$$

$$\leq \sum_{\vec{b_i} \in B_i} \sum_{\pi'_i \in \mathcal{N}(\pi_i^m)} Pr[(Q^{\pi'_i}_{\vec{b_i}} - Q^{\pi_i^m}_{\vec{b_i}}) \geq (R_i(\pi'_i, \pi_j) - R_i(\pi_i^m, \pi_j)) + \epsilon(m, c_i^{\pi'_i,\vec{b_i}}, c_i^{\pi^m,\vec{b_i}}, k_m)]$$

$$\leq \sum_{|\Omega_i|^T} \sum_{\pi'_i \in \mathcal{N}(\pi_i^m)} \exp\{-2i(\frac{\epsilon(m, c_i^{\pi'_i,\vec{b_i}}, c_i^{\pi_i^m,\vec{b_i}})}{\Lambda^{\vec{aj}}(\pi_i, \pi_j)})^2\} = |\Omega_i|^T |\mathcal{N}^<(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2(k_m - 1)|\Omega_i|^T \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|}$$

$$a_{m,\pi_j} = |\mathcal{N}^<(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2(k_m - 1) \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|}$$

When generalizing over all transformations, we get:

$$a^n_m = \sum_{\pi_j} a^n_{m,\pi_j} Pr(\pi_j) \leq \frac{1}{|\Pi_j| \sum_{\pi_j} |\mathcal{N}(\pi_i^m)|} \sum_{\pi_j} |\mathcal{N}_i^<(\pi_i^m)| \frac{\delta_m |\Pi_j|}{2(k_m - 1)}$$

Since MCES-IP reduces to the same error as MCES-P-PALO, using its unique ϵ and k_m, we omit the rest of the proof and refer to the proof for Proposition 1.

□

Proof of Equation (5)

Proof. For the probability of an individual observation sequence, Hoeffding's inequality bounds the probability of an error in sampling. Here, $Pr(\vec{o}|\pi_i)$ represents the sampled likelihood of $\vec{o}$ occurring when agent i follows policy π_i, and $Pr^*(\vec{o}|\pi_i$ is the true probability. Let us first consider the case where $|\vec{o}| = 1$, or the observation sequence is merely a single observation.

$$Pr(|Pr(\vec{o}|\pi_i) - Pr^*(\vec{o}|\pi_i)| > \epsilon_{prune}) \leq 2\exp\{-2k_m\epsilon^2_{prune}\}$$

Since we want to bound the likelihood of any of the observation probabilities being in error, and $Pr(A \vee B \vee \ldots) \leq a + b + \ldots$ where $Pr(A) \leq a, Pr(B) \leq b, \ldots$, then:

$$Pr\left(\sum_{\vec{o} \in \Omega} |Pr(\vec{o}|\pi_i) - Pr^*(\vec{o}|\pi_i)| > |\Omega|\epsilon_{prune}\right) \leq \frac{2|\Omega|}{\exp\{2k_m\epsilon^2_{prune}\}}$$

Probabilities for observation sequences greater than length one involves a summation over a larger set, represented by the Cartesian power of order length equal to the length of the vector. Given some observation sequence length t,

$$Pr\left(\sum_{\vec{o} \in \Omega^t} |Pr(\vec{o}|\pi_i) - Pr^*(\vec{o}|\pi_i)| > |\Omega|^t\epsilon_{prune}\right) \leq \frac{2|\Omega|^t}{\exp\{2k_m\epsilon^2_{prune}\}}$$

For any length of observation sequence, from 0 to the maximum length, $T - 1$,

$$Pr\left(\sum_{t=0}^{T-1} \sum_{\vec{o} \in \Omega^t} |Pr(\vec{o}|\pi_i) - Pr^*(\vec{o}|\pi_i)| > \frac{|\Omega|^T - 1}{|\Omega| - 1}\epsilon_{prune}\right) \leq \frac{2(|\Omega|^T - 1)}{(|\Omega| - 1)\exp\{2k_m\epsilon^2_{prune}\}}$$

□

Proof of Equation (12)

Proof. Pruning for MCES-IP follows similarly from MCES-P-PALO, except our probabilities for observation sequences are based on predicted opponent action sequences as well. This extra dimension increases the amount of observation sequence likelihood probabilities that might be in error.

$$Pr\left(\sum_{t=0}^{T-1}\sum_{\vec{a_j}\in\mathcal{A}^t}\sum_{\vec{o}\in\Omega^t}|Pr(\vec{o}|\pi_i)-Pr^*(\vec{o}|\pi_i)|>\frac{|A|^T|\Omega|^T-1}{|A||\Omega|-1}\epsilon_{prune}\right)\leq\frac{2(|A|^T|\Omega|^T-1)}{(|A||\Omega|-1)\exp\{2k_m\epsilon_{prune}^2}$$

□

Proof of Proposition 5

Proof. We delineate sample bound complexities for MCES-MP-PALO and MCES-FMP-PALO as:

$$k_m^{MP}=\left\lceil 2\left(\frac{\Lambda(\pi,\pi')}{\epsilon}\right)^2\ln\frac{2N^{MP}}{\delta_m}\right\rceil\qquad k_m^{FP}=\left\lceil 2\left(\frac{\Lambda(\pi_i,\pi_i')}{\epsilon}\right)^2\ln\left(\frac{\sqrt[2Z]{4Z-2}N^{FP}}{\sqrt[2Z]{\delta_m}}\right)\right\rceil$$

We derive the condition by which MCES-FMP-PALO exhibits a smaller sample complexity range via simple substitution of the inequality $k_m^{FP}<k_.m^{MP}$

$$\ln\left(\frac{\sqrt[2Z]{4Z-2}N^{FP}}{\sqrt[2Z]{\delta_m}}\right)<\left(\frac{\Lambda(\pi,\pi')}{\Lambda(\pi_i,\pi_i')}\right)^2\ln\frac{2N^{MP}}{\delta_m}$$

π' is the neighbor that satisfies $\max_{\pi'}\Lambda(\pi,\pi')$. For MCESMP +PALO, when using Eq. 1 and 5, $\Lambda(\pi,\pi')=2T\left(\sum_{i\in\mathcal{I}}\mathcal{R}_{max}^i+\mathcal{R}_{max}^G\right)$. MCES-FMP-PALO applies Eq. 1 to 7, defining $\Lambda(\pi_i,\pi_i')=2T\left(\mathcal{R}_{max}^i+\mathcal{R}_{max}^G\right)$. Utilizing these definitions, we can see that MCES-FMP-PALO will always have a smaller sample complexity k_m.

$$\ln\left(\frac{\sqrt[2Z]{4Z-2}N^{FP}}{\sqrt[2Z]{\delta_m}}\right)<\left(1+\frac{\sum_{i\in\bar{\mathcal{I}}}\mathcal{R}_{max}^i}{\mathcal{R}_{max}^i+\mathcal{R}_{max}^G}\right)^2\ln\frac{2N^{MP}}{\delta_m}$$

□

Proof of Proposition 6

Proof. In this proof, we establish bounds for only two agents for the sake of brevity, but it is straightforward to apply it to more agents. We use $\mathcal{N}(\pi)$ as a function to denote the local neighborhood of policies that varies from π by a single action-observation pair. In transformation, the following errors may occur:

(A) After seeing $n<k$ samples, MCES-FMP-PALO transforms joint policy $\{\pi_i^m,\pi_j^m\}$ to neighboring joint policy $\{\pi_i',\pi_j'\}$ where $\pi_i'\in\mathcal{N}(\pi_i^m)$ and $\pi_j'\in\mathcal{N}(\pi_i^m)$, because $\{\pi_i',\pi_j'\}$ appears to be better than $\{\pi_i^m,\pi_j^m\}$, but neither policy is an improvement

(B) As above, but either π_i' or π_j' is not an improvement, but not both

(C) After seeing $n<k$ samples, MCES-FMP-PALO cannot find a better joint policy $\{\pi_i',\pi_j'\}$ where $\pi_i'\in\mathcal{N}(\pi_i^m)$ and $\pi_j'\in\mathcal{N}(\pi_i^m)$ that dominates joint policy $\{\pi_i^m,\pi_j^m\}$, as all π_i' and π_j' appear worse than $\{\pi_i^m,\pi_j^m\}$, but in fact there is a better joint policy

(D) As above, but either π'_i or π'_j would dominate the joint policy

(E) After seeing k samples, MCES-FMP-PALO transforms joint policy $\{\pi_i^m, \pi_j^m\}$ to neighboring joint policy $\{\pi'_i, \pi'_j\}$ where $\pi'_i \in \mathcal{N}(\pi_i^m)$ and $\pi'_j \in \mathcal{N}(\pi_i^m)$, because $\{\pi'_i, \pi'_j\}$ appears to be better than $\{\pi_i^m, \pi_j^m\}$, but neither policy is an improvement

(F) As above, but either π'_i or π'_j is not an improvement, but not both

(G) After seeing k samples, MCES-FMP-PALO cannot find a better joint policy $\{\pi'_i, \pi'_j\}$ where $\pi'_i \in \mathcal{N}(\pi_i^m)$ and $\pi'_j \in \mathcal{N}(\pi_i^m)$ that dominates joint policy $\{\pi_i^m, \pi_j^m\}$, as all π'_i and π'_j appear worse than $\{\pi_i^m, \pi_j^m\}$, but in fact there is a better joint policy

(H) As above, but either π'_i or π'_j would dominate the joint policy

We apply the following to decide when to climb from one individual policy, π, to another, π'.

$$\epsilon(m,p,q) = \begin{cases} \epsilon^*(m,p) & \text{if } p = q < k_m \\ \frac{\epsilon}{2} & \text{if } p = q = k_m \\ +\infty & \text{otherwise} \end{cases}$$

where,

$$\epsilon^*(m,p) = \Lambda(\pi,\pi')\sqrt{\frac{1}{2p}\ln\left(\frac{\sqrt[4]{6(k_m-1)}N}{\sqrt[4]{\delta_m}}\right)}$$

and,

$$k_m = \left\lceil 2(\frac{\Lambda(\pi,\pi')}{\epsilon})^2 \ln(\frac{\sqrt[4]{6}N^{FP}}{\sqrt[4]{\delta_m}}) \right\rceil$$

Λ is defined by implementing Eq. 1 in 7 and $N = |\mathcal{N}(\pi^m)|$ is the size of the local neighborhood. Let $\vec{\pi} = \langle \pi_i, \pi_j \rangle$. Let $E^i[\vec{\pi}]$ represent the true value, as opposed to empirically sampled value Q, of following a policy π_i when the other agent is following π_j, and $E^j[\vec{\pi}]$ analogously for the other agent. We can represent each of the previously mentioned types of error in MCES-FMP-PALO with the following expressions.

$$a_m^n = Pr[\exists\{\pi'_i,\pi'_j\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi'_i} - Q_{\pi_i^m}) \geq \epsilon(m, c_i^{\pi'_i}, c_i^{\pi_i^m}),$$
$$(Q_{\pi'_j} - Q_{\pi_j^m}) \geq \epsilon(m, c_j^{\pi'_j}, c_j^{\pi_j^m}), \text{and } E^i[\vec{\pi}'] < E^i[\vec{\pi}^m], E^j[\vec{\pi}'] < E^j[\vec{\pi}^m]]$$

$$b_m^n = Pr[\exists\{\pi'_i,\pi'_j\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi'_i} - Q_{\pi_i^m}) \geq \epsilon(m, c_i^{\pi'_i}, c_i^{\pi_i^m}),$$
$$(Q_{\pi'_j} - Q_{\pi_j^m}) \geq \epsilon(m, c_j^{\pi'_j}, c_j^{\pi_j^m}), \text{and } [(E^i[\vec{\pi}'] < E^i[\vec{\pi}^m], E^j[\vec{\pi}'] > E^j[\vec{\pi}^m] + \epsilon)$$
$$\text{or } (E^i[\vec{\pi}'] > E^i[\vec{\pi}^m] + \epsilon, E^j[\vec{\pi}'] < E^j[\vec{\pi}^m])]]$$

$$c_m^n = Pr[\exists\{\pi_i', \pi_j'\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi_i'} - Q_{\pi_i^m}) < \epsilon - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m}),$$
$$(Q_{\pi_j'} - Q_{\pi_j^m}) < \epsilon - \epsilon(m, c_j^{\pi_j'}, c_j^{\pi_j^m}), \text{and } E^i[\vec{\pi}'] > E^i[\vec{\pi}^m] + \epsilon, E^j[\vec{\pi}'] > E^j[\vec{\pi}^m] + \epsilon]$$

$$d_m^n = Pr[\exists\{\pi_i', \pi_j'\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi_i'} - Q_{\pi_i^m}) < \epsilon - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m}),$$
$$(Q_{\pi_j'} - Q_{\pi_j^m}) < \epsilon - \epsilon(m, c_j^{\pi_j'}, c_j^{\pi_j^m}), \text{and } [(E^i[\vec{\pi}'] > E^i[\vec{\pi}^m] + \epsilon, E^j[\vec{\pi}'] < C(\pi_j^m), \pi_i^m)$$
$$\text{or } (E^i[\vec{\pi}'] < E^i[\vec{\pi}^m], E^j[\vec{\pi}'] > E^j[\vec{\pi}^m] + \epsilon)]]$$

$$e_m = Pr[\exists\{\pi_i', \pi_j'\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi_i'} - Q_{\pi_i^m}) \geq \frac{\epsilon}{2}, (Q_{\pi_j'} - Q_{\pi_j^m}) \geq \frac{\epsilon}{2},$$
$$\text{and } E^i[\vec{\pi}'] < E^i[\vec{\pi}^m], E^j[\vec{\pi}'] < E^j[\vec{\pi}^m]]$$

$$f_m = Pr[\exists\{\pi_i', \pi_j'\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi_i'} - Q_{\pi_i^m}) \geq \frac{\epsilon}{2}, (Q_{\pi_j'} - Q_{\pi_j^m}) \geq \frac{\epsilon}{2},$$
$$\text{and } [(E^i[\vec{\pi}'] < E^i[\vec{\pi}^m], E^j[\vec{\pi}'] > E^j[\vec{\pi}^m] + \epsilon)$$
$$\text{or } (E^i[\vec{\pi}'] > E^i[\vec{\pi}^m] + \epsilon, E^j[\vec{\pi}'] < E^j[\vec{\pi}^m])]]$$

$$g_m = Pr[\exists\{\pi_i', \pi_j'\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi_i'} - Q_{\pi_i^m}) < \frac{\epsilon}{2}, (Q_{\pi_j'} - Q_{\pi_j^m}) < \frac{\epsilon}{2},$$
$$\text{and } E^i[\vec{\pi}'] > E^i[\vec{\pi}^m] + \epsilon, E^j[\vec{\pi}'] > E^j[\vec{\pi}^m] + \epsilon]$$

$$h_m = Pr[\exists\{\pi_i', \pi_j'\} \in \mathcal{N}(\pi_i^m) \times \mathcal{N}(\pi_j^m) : (Q_{\pi_i'} - Q_{\pi_i^m}) < \frac{\epsilon}{2}, (Q_{\pi_j'} - Q_{\pi_j^m}) < \frac{\epsilon}{2},$$
$$\text{and } [(E^i[\vec{\pi}'] > E^i[\vec{\pi}^m] + \epsilon, E^j[\vec{\pi}'] < C(\pi_j^m), \pi_i^m)$$
$$\text{or } (E^i[\vec{\pi}'] < E^i[\vec{\pi}^m], E^j[\vec{\pi}'] > E^j[\vec{\pi}^m] + \epsilon)]]$$

We represent the existential quantifiers above as finite disjunctions over the elements of the joint local neighborhood, $\mathcal{N}(\vec{\pi})$. For the individual agent, considering a set of neighboring policies that are truly worse than the current policy given the other agent is fixed can be represented as: $\mathcal{N}_i^<(\vec{\pi}^m) = \{\pi_i' \in \mathcal{N}(\pi_i^m) | E^i[\pi_i', \pi_j^m] < E^i[\pi_i^m, \pi_j^m]\}$, and similarly for agent j. Subsequently, we represent the neighborhood of joint policies where both agents transformed policies are truly worse as $\mathcal{N}^{<<}(\vec{\pi}^m) = \{\vec{\pi}' \in \mathcal{N}(\vec{\pi}^m) | E^i[\vec{\pi}'] < E^i[\vec{\pi}^m] \wedge E^j[\vec{\pi}'] < E^j[\vec{\pi}^m]\}$. We omit the parameter as it is clear from context, resulting in $\mathcal{N}^{<<}$. Note that $|\mathcal{N}^{<<}| = |\mathcal{N}_i^<| \cdot |\mathcal{N}_j^<|$.

$$a_m^n \leq \sum_{\vec{\pi}' \in \mathcal{N}^{<<}} Pr[(Q_{\pi_i'} - Q_{\pi_i^m}) \geq \epsilon(m, c_j^{\pi_i'}, c_i^{\pi_i^m}) \wedge (Q_{\pi_j'} - Q_{\pi_j^m}) \geq \epsilon(m, c_j^{\pi_j'}, c_j^{\pi_j^m})]$$
$$\leq \sum_{\vec{\pi}' \in \mathcal{N}^{<<}} Pr[(Q_{\pi_i'} - Q_{\pi_i^m}) \geq (E^i[\vec{\pi}'] - E^i[\vec{\pi}^m]) + \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m})] \cdot$$
$$Pr[(Q_{\pi_j'} - Q_{\pi_j^m}) \geq (E^j[\vec{\pi}'] - E^j[\vec{\pi}^m]) + \epsilon(m, c_j^{\pi_j'}, c_j^{\pi_j^m})]$$

$$\leq \sum_{\vec{\pi}' \in \mathcal{N}^{<<}} \exp\{-2p[(\frac{\epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m})}{\Lambda(\pi_i, \pi_i')})^2 + (\frac{\epsilon(m, c_j^{\pi_j'}, c_j^{\pi_j^m})}{\Lambda(\pi_j, \pi_j')})^2]\} \tag{23}$$

$$= |\mathcal{N}^{<<}| \frac{\sqrt{\delta_m}}{\sqrt{6}\sqrt{k_m - 1}|\mathcal{N}(\pi_i^m)|} \cdot \frac{\sqrt{\delta_m}}{\sqrt{6}\sqrt{k_m - 1}|\mathcal{N}(\pi_j^m)|} = \frac{|\mathcal{N}^{<<}|\delta_m}{6(k_m - 1)|\mathcal{N}(\vec{\pi}^m)|}$$

Line 23 follows from Hoeffding's Inequality, where $(Q_{\pi_i'} - Q_{\pi_i^m})$ is the sample average approximating $E^i[\vec{\pi}'] - E^i[\vec{\pi}^m]$. 23 is reduced when satisfying the condition $p = q < k_m$. We can also express probabilities for sets of policies that are truly greater as well, using $\mathcal{N}_i^{>}(\vec{\pi}^m) = \{\pi_i' \in \mathcal{N}(\pi_i^m) | C(\pi_i', \pi_j^m) > C(\pi_i^m, \pi_j^m) + \epsilon\}$. Again, this is written similarly for agent j. The joint notation follows identically as above. B is then computed similarly, but the space of possible neighbors is over the union of $\mathcal{N}^{<>}$ and $\mathcal{N}^{><}$.

$$b_m^n \leq \frac{(|\mathcal{N}^{<>}| + |\mathcal{N}^{><}|)\delta_m}{6(k_m - 1)|\mathcal{N}(\vec{\pi}^m)|}$$

C follows from A, but in the case where MCES-FMP-PALO terminates without transformation.

$$c_m^n \leq \sum_{\vec{\pi}' \in \mathcal{N}^{>>}} Pr[(Q_{\pi_i'} - Q_{\pi_i^m}) \leq \epsilon - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m}) \wedge (Q_{\pi_j'} - Q_{\pi_j^m}) \leq \epsilon - \epsilon(m, c_j^{\pi_j'}, c_j^{\pi_j^m})]$$

$$\leq \sum_{\vec{\pi}' \in \mathcal{N}^{>>}} Pr[(Q_{\pi_i'} - Q_{\pi_i^m}) < (E^i[\vec{\pi}'] - E^i[\vec{\pi}^m]) - \epsilon(m, c_i^{\pi_i'}, c_i^{\pi_i^m})]\cdot$$

$$Pr[(Q_{\pi_j'} - Q_{\pi_j^m}) < (E^j[\vec{\pi}'] - E^j[\vec{\pi}^m]) - \epsilon(m, c_j^{\pi_j'}, c_j^{\pi_j^m})] = \frac{|\mathcal{N}^{>>}|\delta_m}{6(k_m - 1)|\mathcal{N}(\vec{\pi}^m)|} \tag{24}$$

Line 24 follows analogously from A. As with B, D just expands the neighborhood.

$$d_m^n \leq \frac{(|\mathcal{N}^{><}| + |\mathcal{N}^{<>}|)\delta_m}{6(k_m - 1)|\mathcal{N}(\vec{\pi}^m)|}$$

We can bound at $p = k_m$ samples in the same fashion.

$$e_m \leq \sum_{\vec{\pi}' \in \mathcal{N}^{<<}} Pr[(Q_{\pi_i'} - Q_{\pi_i^m}) \geq \frac{\epsilon}{2} \wedge (Q_{\pi_j'} - Q_{\pi_j^m}) \geq \frac{\epsilon}{2}]$$

$$\leq \sum_{\vec{\pi}' \in \mathcal{N}^{<<}} \exp\left\{-2k_m \left[\left(\frac{\epsilon/2}{\Lambda(\pi_i, \pi_i')}\right)^2 + \left(\frac{\epsilon/2}{\Lambda(\pi_j, \pi_j')}\right)^2\right]\right\} = \frac{|\mathcal{N}^{<<}|\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|} \tag{25}$$

where Line 25 is obtained by replacing k_m with its derived value. F is obtained as in B with the reduction in Line 25.

$$f_m \leq \sum_{\vec{\pi}' \in \mathcal{N}^{<>} \cup \mathcal{N}^{><}} Pr[(Q_{\pi_i'} - Q_{\pi_i^m}) \geq \frac{\epsilon}{2} \wedge (Q_{\pi_j'} - Q_{\pi_j^m}) \geq \frac{\epsilon}{2}] \leq \frac{(|\mathcal{N}^{<>}| + |\mathcal{N}^{><}|)\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|}$$

G and H are solved in the same fashion as C and D.

$$g_m \leq \sum_{\vec{\pi}' \in \mathcal{N}^{>>}} Pr\left[(Q_{\pi_i'} - Q_{\pi_i^m}) < \frac{\epsilon}{2} \wedge (Q_{\pi_j'} - Q_{\pi_j^m}) < \frac{\epsilon}{2}\right] \leq \frac{|\mathcal{N}^{>>}|\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|}$$

$$h_m \leq \sum_{\vec{\pi}' \in \mathcal{N}^{><} \cup \mathcal{N}^{<>}} Pr[(Q_{\pi_i'} - Q_{\pi_i^m}) < \frac{\epsilon}{2} \wedge (Q_{\pi_j'} - Q_{\pi_j^m}) < \frac{\epsilon}{2}] \leq \frac{(|\mathcal{N}^{><}| + |\mathcal{N}^{<>}|)\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|}$$

Then, the probability that MCES-FMP-PALO makes an error after n transformations:

$$\sum_{n=1}^{k_m}[a_m^n + b_m^n + c_m^n + d_m^n] + e_m + f_m + g_m + h_m$$

$$\leq [\frac{(k_m-1)\delta_m}{6(k_m-1)|\mathcal{N}(\vec{\pi}^m)|}(|\mathcal{N}^{<<}| + |\mathcal{N}^{<>}| + |\mathcal{N}^{><}| + |\mathcal{N}^{>>}| + |\mathcal{N}^{><}| + |\mathcal{N}^{<>}|)] +$$

$$\frac{\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|}(|\mathcal{N}^{<<}| + |\mathcal{N}^{<>}| + |\mathcal{N}^{><}| + |\mathcal{N}^{>>}| + |\mathcal{N}^{><}| + |\mathcal{N}^{<>}|)$$

$$\leq \left[\frac{\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|}(|N(\vec{\pi}^m)| + |\mathcal{N}^{<>}| + |\mathcal{N}^{><}|)\right] + \frac{\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|}(|\mathcal{N}(\vec{\pi}^m)| + |\mathcal{N}^{<>}| + |\mathcal{N}^{><}|) \tag{26}$$

$$\leq 2\left[\frac{\delta_m}{6|\mathcal{N}(\vec{\pi}^m)|}(3|N(\vec{\pi}^m)|\right] = \delta_m \tag{27}$$

Line 26 follows from the observations that $\mathcal{N}^{**} = \mathcal{N}_i^* \times \mathcal{N}_j^*$ and $|\mathcal{N}_i^<| + |\mathcal{N}_i^>| \leq |\mathcal{N}_i|$. Line 27 follows from the observation that any product of subsets of $\mathcal{N}$ is bounded by the entire set (e.g. $|\mathcal{N}_i^{<<}| \leq |\mathcal{N}|$). Then, over all mutations, δ_m is bound by δ.

$$\sum_{m=1}^{\infty}[\sum_{n=1}^{k}[a_m^n + b_m^n + c_m^n + d_m^n] + e_m + f_m + g_m + h_m] \leq \sum_{m=1}^{\infty}\delta_m = \sum_{m=1}^{\infty}\frac{6\delta}{m^2\pi^2} = \frac{6\delta}{\pi^2}\frac{\pi^2}{6} = \delta$$

□