Efficient Planning in Combinatorial Action Spaces with Applications to Cooperative Multi-Agent Reinforcement Learning

Volodymyr Tkachuk[*]
University of Alberta

Seyed Alireza Bakhtiari[*]
University of Alberta

Johannes Kirschner
University of Alberta

Matej Jusup
ETH Zurich

Ilija Bogunovic
University College London

Csaba Szepesvári
University of Alberta/DeepMind

Abstract

A practical challenge in reinforcement learning are combinatorial action spaces that make planning computationally demanding. For example, in cooperative multi-agent reinforcement learning, a potentially large number of agents jointly optimize a global reward function, which leads to a combinatorial blow-up in the action space by the number of agents. As a minimal requirement, we assume access to an argmax oracle that allows to efficiently compute the greedy policy for any Q-function in the model class. Building on recent work in planning with local access to a simulator and linear function approximation, we propose efficient algorithms for this setting that lead to polynomial compute and query complexity in all relevant problem parameters. For the special case where the feature decomposition is additive, we further improve the bounds and extend the results to the kernelized setting with an efficient algorithm.

1 INTRODUCTION

Reinforcement learning (RL) is concerned with training data-driven agents to make optimal decisions in interactive environments. An agent interacts with an environment by choosing actions and observing its state and a reward signal. The goal is to learn an optimal policy that maximizes the total reward. Efficiently computing optimal policies, also known as *planning*, is therefore at the heart of any reinforcement learning algorithm.

Recent works have successfully applied reinforcement learning algorithms to complex domains including video games (Mnih et al., 2013), tokamak plasmas control (Degrave et al., 2022), robotic manipulation tasks (Akkaya et al., 2019), to name a few. A common theme of these works is that the agent is trained on a simulated environment. This provides additional flexibility on how the agent can interact with the environment. A reasonable assumption is that the internal state of the simulator can be saved ('checkpointing') and later revisited.

In this work, we formally study *efficient planning with local access to a simulator*. The local access model was recently proposed by Yin et al. (2021) with the goal of making the simulation access model more practical in applications. Local access means that the only states at which the planner can query the simulator are the initial state or states returned in response to previously issued queries. Efficient planning means that given an initial state, the learner outputs a near-optimal policy using polynomial compute and queries in all relevant parameters.

Motivated by the increasing complexity of applications, we specifically study the case where the state space is large or continuous. To avoid the query complexity scaling with the size of the state space, it is standard to introduce linear function approximation (e.g., Bertsekas and Ioffe, 1996; Lagoudakis and Parr, 2003; Munos, 2005; Lattimore et al., 2020). In particular, we assume linear ϵ-realizability of joint state-action value functions for *all* policies. This is motivated by the recent realization that realizability of the optimal state-action value function alone is not sufficient to develop a query-efficient planner (Weisz et al., 2021). However, even under stronger realizability assumptions, previous approaches are not computationally efficient in the case where the action space is *combinatorially large*, and direct enumeration of the action space becomes infeasible. Therefore, we work with a minimal oracle assumption that allows us to compute the greedy policy for any Q-function in the model class (which amounts to solving a linear optimization over the feature space).

One prominent special case of this setting is multi-

[*] Equal contribution

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

	Algorithms	Query $\epsilon = 0$	Query $\epsilon > 0$	Sub-optimality $\epsilon > 0$	Computation
LSPI	NAIVE	$\tilde{\mathcal{O}}\left(\frac{d^3}{\kappa^2(1-\gamma)^8}\right)$	$\tilde{\mathcal{O}}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{d}}{(1-\gamma)^2}\right)$	poly$(\lvert\mathcal{A}\rvert, d)$
	EGSS	$\tilde{\mathcal{O}}\left(\frac{d^{3+\mathbf{1}}}{\kappa^2(1-\gamma)^8}\right)$	$\tilde{\mathcal{O}}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{d}\boldsymbol{\sqrt{d}}}{(1-\gamma)^2}\right)$	poly(d)
	DAV	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m^2}d^3}{\kappa^2(1-\gamma)^8}\right)$	$\tilde{\mathcal{O}}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{d}\boldsymbol{m}}{(1-\gamma)^2}\right)$	poly$(\sum_{i=1}^m \lvert\mathcal{A}^{(i)}\rvert, d)$
	KERNEL-DAV	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m^2}\tilde{\Gamma}^3}{\kappa^2(1-\gamma)^8}\right)$	$\tilde{\mathcal{O}}\left(\frac{\tilde{\Gamma}^2}{\epsilon^2(1-\gamma)^4}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{\tilde{\Gamma}}\boldsymbol{m}}{(1-\gamma)^2}\right)$	poly$(\sum_{i=1}^m \lvert\mathcal{A}^{(i)}\rvert, \tilde{\Gamma})$
POLITEX	NAIVE	$\tilde{\mathcal{O}}\left(\frac{d^3}{\kappa^4(1-\gamma)^9}\right)$	$\tilde{\mathcal{O}}\left(\frac{d}{\epsilon^4(1-\gamma)^5}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{d}}{(1-\gamma)}\right)$	poly$(\lvert\mathcal{A}\rvert, d)$
	EGSS	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m}d^{3+\mathbf{1}}}{\kappa^4(1-\gamma)^9}\right)$	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m}d}{\epsilon^4(1-\gamma)^5}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{d}\boldsymbol{\sqrt{d}}}{(1-\gamma)}\right)$	poly$(\sum_{i=1}^m \lvert\mathcal{A}^{(i)}\rvert, d)$
	DAV	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m^3}d^3}{\kappa^4(1-\gamma)^9}\right)$	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m}d}{\epsilon^4(1-\gamma)^5}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{d}\boldsymbol{m}}{(1-\gamma)}\right)$	poly$(\sum_{i=1}^m \lvert\mathcal{A}^{(i)}\rvert, d)$
	KERNEL-DAV	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m^2}\tilde{\Gamma}^3}{\kappa^2(1-\gamma)^8}\right)$	$\tilde{\mathcal{O}}\left(\frac{\boldsymbol{m}\tilde{\Gamma}}{\epsilon^4(1-\gamma)^5}\right)$	$\tilde{\mathcal{O}}\left(\frac{\epsilon\sqrt{\tilde{\Gamma}}\boldsymbol{m}}{(1-\gamma)}\right)$	poly$(\sum_{i=1}^m \lvert\mathcal{A}^{(i)}\rvert, \tilde{\Gamma})$

Table 1: Query complexity and sub-optimality bounds of algorithms proposed in Section 4 and 5 in the realizable ($\epsilon = 0$) and ϵ-misspecified ($\epsilon > 0$) setting. NAIVE refers to a direct implementation of the approach by Yin et al. (2021). For $\epsilon = 0$,the sub-optimality gap is $\kappa > 0$, while for $\epsilon > 0$, the sub-optimality gap is given in the third column. All algorithms require $\mathcal{O}(\text{poly}(\frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}), \log(b), \frac{1}{\epsilon}))$ computation. LSPI-EGSS requires access to a 'greedy oracle' (Assumption 3). Results for LSPI-(KERNEL-)DAV and POLITEX hold for product action sets $\mathcal{A} = \mathcal{A}^{(1)} \times \cdots \times \mathcal{A}^{(m)}$ and Assumption 4.

agent reinforcement learning. Multi-agent reinforcement learning has been a recent research focus with multiple promising attempts at tackling complex multi-agent problems, e.g., team games (Baker et al., 2019), large scale traffic signal control (Chu et al., 2019), cooperative controls in powergrids (Chen et al., 2021a) among others. Naively applying single-agent planning algorithms fails to achieve efficiency in the multi-agent setting because the single-agent algorithms typically face an exponential blow-up of the action space in the number agents. In many practical tasks, however, there is an inherent structure in the underlying dynamics that can be exploited to address both efficiency and scalability issues.

Contributions Our first contribution is a novel oracle-efficient variant of the Confident Monte-Carlo least-squares policy iteration (CONFIDENT MC-LSPI) algorithm by Yin et al. (2021), for combinatorially large action spaces. The key insight is an efficient implementation of the *uncertainty check*, that determines the diversity of the state and action set used for estimation. We also study a special case where the Q-function has an additive structure in the features (formally introduced in Assumption 4), which leads to improved bounds in the regime where the dimension is large. In the multi-agent setting, the decomposition corresponds to agent-specific features, and the proposed algorithms achieve polynomial compute and query complexity in the number of agents and other quantities of interest. We further introduce a kernelized variant, which under the corresponding additivity assumption admits an efficient implementation. Lastly, the additive structure leads to an efficient implementation of the CONFIDENT MC-POLITEX algorithm that admits improved bounds in the misspecified setting. The formal results are summarized in Table 1.

2 RELATED WORK

Computing optimal policies, also known as *planning*, is a central challenge in reinforcement learning (Sutton and Barto, 2018; Szepesvári, 2010). The two most classical planning algorithms are value iteration (Bellman, 1957) and policy iteration (Howard, 1960). Approximate versions of value and policy iteration were analyzed by Munos (2003, 2005); Farahmand et al. (2010). A common setting is planning with a *generative model* (also *global* simulator access), where the learner can query the transition kernel at any state and action (Kakade, 2003). In the corresponding tabular setting the query complexity of value and policy iteration are completely understood (e.g., Azar et al., 2012; Gheshlaghi Azar et al., 2013). When combined with function approximation, the picture becomes more nuanced. A lower bound under misspecification was provided by Du et al. (2019). Sample complexity bounds for least-squares policy iteration (Bertsekas and Ioffe, 1996; Lagoudakis and Parr, 2003) are by Lattimore et al. (2020). The latter work combines a G-experimental design over state-action pairs with Monte-Carlo rollouts to obtain value estimates for the policies. In similar fashion, least-squares value iteration (LSVI) was analyzed in the generative model setting (Agarwal et al., 2020a). Yet another approach is Politex (Abbasi-Yadkori et al., 2019; Szepesvári, 2022a), which uses mirror descent to improve the policy.

A much larger body of work focuses on the online setting, where the learner interacts with the environment in one or multiple episodes. Early work that uses function approximation includes (Bradtke and Barto, 1996; Melo and Ribeiro, 2007). Recent works provide query complexity guarantees under various models (Osband et al., 2016; Yang et al., 2020; Ayoub et al., 2020; Zanette et al.,

2020; Du et al., 2021; Zhou et al., 2021). This includes approaches that are computationally efficient for small action sets (Jin et al., 2020; Agarwal et al., 2020b). We are not aware of provably query efficient algorithms with *only* linear Q_π-realizability (Assumption 1) for the online setting. Abbasi-Yadkori et al. (2019); Lazic et al. (2021); Wei et al. (2021) prove bounds with a *feature excitation* condition, although these works do not consider large action sets. Negative results under weaker assumptions are known, e.g. for Q^*-realizability (Weisz et al., 2021) and approximate Q_π-realizability (Du et al., 2019).

Recently, Yin et al. (2021) introduced the *local access* model, in which the learner can query the simulator at the initial state or any state observed during planning. They further introduce a Monte-Carlo policy iteration algorithm that provides the basis of our work. Different to this previous work, we consider the combinatorial action set setting, and provide new algorithms that avoid scaling of the computational complexity with the size of the action set. Least-squares value iteration with local access was analyzed by Hao et al. (2022). For a detailed discussion on different simulators models we refer the reader to (Yin et al., 2021).

Relatively few related works on computationally efficient planning in MDPs are concerned with combinatorial action spaces. This topic has received attention in the context of factored MDPs in planning (Dean et al., 1998; Geißer et al., 2020; Raghavan et al., 2012), online RL (Osband and Van Roy, 2014; Xu and Tewari, 2020; Tian et al., 2020; Chen et al., 2020) and in the empirical literature (Delarue et al., 2020; Hubert et al., 2021) with applications to vehicle routing and control problems. We are not aware of prior work with query complexity guarantee for MDPs with large action sets, however there is a long line of works on combinatorial bandits (e.g., Cesa-Bianchi and Lugosi, 2012; Chen et al., 2013; Shleyfman et al., 2014; Combes et al., 2015; Jourdan et al., 2021). Relevant in this context are also kernelized bandit algorithms (Bayesian optimization) that exploit additive structure of the reward function (Kandasamy et al., 2015; Wang et al., 2019; Kirschner and Krause, 2021; Mutny and Krause, 2018; Rolland et al., 2018). We consider a similar assumption in Section 5 as a special case.

Multi-agent reinforcement learning (Busoniu et al., 2008; Zhang et al., 2021) can be understood as a combinatorial setting, which has a large body of works on its own. Query complexity bounds focus mostly on the competitive setting, e.g. in tabular Markov games (e.g., Shapley (1953); Song et al. (2021); Tian et al. (2021); Bai and Jin (2020); Liu et al. (2021); Leonardos et al. (2021)). One of the key challenges is the exponential blowup in the action space with the number of agents, which is sometimes refered to as 'curse of multi-agents'. Jin et al. (2021) introduce a computationally efficient algorithm for tabular Markov games. Multi-agent reinforcement learning with function approximation is studied by Huang et al. (2021a); Chen et al. (2021b); Jin et al. (2020). These works consider the competitive setting and focus on obtaining query efficient algorithms, while the approaches are not computationally tractable. In the limit where the number of agents becomes large, previous work uses mean-field approximations (Yang et al., 2018; Pasztor et al., 2021).

Most closely related is *cooperative* multi-agent learning. Early work by Guestrin et al. (2001) proposes the use of factored MDPs to make planning tractable via message passing algorithms. Rashid et al. (2018) propose a neural network architecture that allows to decouple the agent rewards in a way such that the greedy policy can be computed efficiently. The goal of these works is to ensure the greedy policy can be computed efficiently. Zohar et al. (2021) consider a setting where a graph structure captures the reward dependencies across the agents; however the guarantees they provide apply only to the bandit setting.

3 PRELIMINARIES

We consider reinforcement learning in an infinite-horizon Markov decision process (MDP) specified by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \gamma)$. As usual, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, and $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition kernel, where $\Delta_{\mathcal{S}}$ denotes the set of probability measures over $\mathcal{S}$. Given a state $s \in \mathcal{S}$ and action vector $a \in \mathcal{A}$, the system transits to a new state $s' \sim \mathbb{P}(s, a)$. The reward function is $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and $\gamma \in [0, 1)$ is the discount factor.

A stationary policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ maps states to a distribution over $\mathcal{A}$. The state value function $V_\pi : \mathcal{S} \to \mathbb{R}$ of a policy π from a state $s \in \mathcal{S}$ is

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \Big| s_0 = s \right].$$

The expectation is over the sequence of states $(s_t)_{t\in\mathbb{N}}$ and actions $(a_t)_{t\in\mathbb{N}}$ queried from the transition kernel $\mathbb{P}$ and the policy π. A policy π^* is *optimal* if $V_{\pi^*} = \max_\pi V_\pi$.

The Q-function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ of a policy π is defined for $s \in \mathcal{S}$ and $a \in \mathcal{A}$ as

$$Q_\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(s,a)} \left[V_\pi(s') \right].$$

In the following we assume that we are given a state-action feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, that allows to approximate the Q-function of any policy as a linear function.

Assumption 1 (Linear Q_π-realizability)**.** *For each policy π there exists a weight vector $w_\pi \in \mathbb{R}^d$, $\|w_\pi\|_2 \le b$ satisfying $\max_{s,a} |Q_\pi(s, a) - w_\pi^\top \phi(s, a)| \le \epsilon$.*

The assumption is commonly used in combination with policy iteration algorithms (Lattimore et al., 2020; Zanette et al., 2020). In particular, the assumption allows to obtain

query complexity results that are independent of the number of states and actions. We remark that the linear MDP assumption (Jin et al., 2020) implies Q_π-realizability, but not vice versa. We also make the following standard boundedness assumption:

Assumption 2 (Bounded features)**.** *We assume that* $\|\phi(s,a)\|_2 \leq 1$ *for all* $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Our main objective is to obtain query and computationally efficient algorithms for the case where the action set $\mathcal{A}$ is *combinatorially* large, and direct enumeration becomes infeasible. To obtain meaningful results in this setting, we assume that the *offline problem* of computing the greedy policy given a *fixed* approximator $w \in \mathbb{R}^d$ can be solved efficiently. This is formally captured in the next assumption.

Assumption 3 (Greedy oracle)**.** *We have access to an oracle* $\mathcal{G}$ *which takes as input a vector* $w \in \mathbb{R}^d$*, a state* $s \in \mathcal{S}$ *and a feature function* $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ *and returns an action that maximizes* $w^\top \phi(s,a)$*. Formally*

$$\mathcal{G}(w,\phi) = \arg\max_{a \in \mathcal{A}} w^\top \phi(s,a)\,,$$

with ties broken arbitrarily.

Combined with the linear Q_π-realizability (Assumption 1), the greedy oracle amounts to solving a *linear* optimization over the action set $\mathcal{A}$. This is a reasonable assumption, as optimized solvers are available for many settings. It is also a *minimal* assumption in the sense that it is required to implement a policy iteration procedure. Note that the assumption can be relaxed to require only an ϵ-approximate solution, which is essentially equivalent to misspecification (Assumption 1). In Section 5 we provide an additive model where the oracle can be directly implemented.

Our goal is to find a computational and query efficient algorithm that given a starting state $\rho \in \mathcal{S}$ returns a κ-optimal policy $\hat{\pi}$, i.e. $V_{\pi^*}(\rho) - V_{\hat{\pi}}(\rho) \leq \kappa$ for $\kappa > 0$ while minimizing the number of queries needed. To obtain queries, the learner is given *local access* to a simulator of the MDP (Yin et al., 2021). A simulator of the MDP takes as input a state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$ and returns a next state $s' \sim \mathbb{P}(s,a)$ and reward $r(s,a)$. A local access simulator restricts the input state $s \in \mathcal{S}$ only to those states which have been visited previously.

An important example where the action set is typically large is cooperative multi-agent reinforcement learning.

Example 1 (Cooperative multi-agent RL)**.** *In the multi-agent setting,* $m \in \mathbb{N}$ *agents act jointly on the MDP* $\mathcal{M}$*. Each agent* $i \in [m]$ *has a set of actions* $\mathcal{A}^{(i)}$ *available where* $[m] := \{1,\dots,m\}$*. We denote the joint action set by* $\mathcal{A} = \mathcal{A}^{(1:m)} := \mathcal{A}^{(1)} \times \dots \times \mathcal{A}^{(m)}$*. The state space* $\mathcal{S}$ *is joint for all agents. A centralized, stationary policy* $\pi : \mathcal{S} \to \Delta_{\mathcal{A}^{(1:m)}}$ *maps states to a distribution over* $\mathcal{A}^{(1:m)}$*. In the* cooperative *setting, the agents jointly maximize a global reward function* $r : \mathcal{S} \times \mathcal{A}^{(1:m)} \to [0,1]$.

Note that the size of the joint action set is exponential in the number of agents, which makes approaches designed for the single agent setting computationally intractable. We will revisit this example in Section 5 where we discuss how an additive feature decomposition leads to algorithms that scale polynomially in the number of agents m. We remark that prior work on multi-agent RL has focused on architectures where the greedy policy can be computed efficiently (e.g., Guestrin et al., 2001; Rashid et al., 2018; Delarue et al., 2020; Zohar et al., 2021).

4 EFFICIENT MC-LSPI

In this section, we extend the CONFIDENT MC-LSPI algorithm proposed by Yin et al. (2021) to the combinatorial action setting. More precisely, Algorithm 1 with Algorithm 5 used for the UNCERTAINTYCHECK is equivalent to the CONFIDENT MC-LSPI algorithm presented in Yin et al. (2021), which relies on either enumerating the action set or solving a quadratic maximization problem, both which become infeasible for large $\mathcal{A}$ in general (e.g., Bhattiprolu et al., 2021). The main challenge is to come up with a procedure that uses only polynomially many calls to the greedy oracle while also scaling polynomially in all other quantities of interest.

At a high level, Algorithm 1 alternates between policy evaluation and policy improvement. For evaluation, a core set is constructed that holds a small but sufficiently diverse set of features corresponding to state-action pairs. For each element of the core set, the ROLLOUT routine (Algorithm 2) returns a Monte-Carlo estimate of the Q-value. During each rollout, the UNCERTAINTYCHECK subroutine (Algorithm 3) determines if a feature should be added to the core set. This procedure is repeated until no more elements are added to the core set. The Monte-Carlo returns from the rollouts are then used to construct a least-squares estimate of $Q_\pi(s,a)$, which in turn is used to improve the policy.

Formally, the outer loop aims to complete K iterations of policy iteration. The goal of each iteration k is to estimate $Q_{\pi_{k-1}}$ using a weight vector $w_k \in \mathbb{R}^d$ and derive a new greedy policy π_k, w.r.t. w_k. For estimation, the algorithm maintains a *core set* $\mathcal{C}$ with elements corresponding to state-action pairs. The elements of the core set $z = (z_s, z_a, z_\phi, z_q) \in \mathcal{C}$ are tuples containing a state $z_s \in \mathcal{S}$, an action $z_a \in \mathcal{A}$, the corresponding feature $z_\phi \in \mathbb{R}^d$, and a value estimate $z_q \in \mathbb{R} \cup \{\text{NONE}\}$. We denote the vector of all value estimates in the core set as $q_\mathcal{C} = (z_q)_{z \in \mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}$. The weight vector w_k to estimate $Q_{\pi_{k-1}}$ is computed using regularized least squares, with $q_\mathcal{C}$ as the targets (line 16). An improved policy based on w_k is then calculated by following the greedy policy with respect to $w^\top \phi(s,a)$ (line 17). The core set is initialized in lines 3-8 by adding the initial state with a *default action* $\bar{a}$, so that there is at least one element in the core set to rollout from (line 3). Then we continuously run the UNCERTAINTYCHECK al-

Algorithm 1 CONFIDENT MC-LSPI

1: **Input:** initial state ρ, initial policy π_0, number of iterations K, threshold τ, number of rollouts n, length of rollout H
2: **Globals:** default action $\bar{a}$, regularization coefficient λ, discount γ, subroutine UNCERTAINTYCHECK
3: $\mathcal{C} \leftarrow \{(\rho, \bar{a}, \phi(\rho, \bar{a}), \text{NONE})\}$
4: status, result $\leftarrow$ UNCERTAINTYCHECK$(\rho, \mathcal{C}, \tau)$
5: **while** status $=$ UNCERTAIN **do**
6: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\text{result}\}$
7: $\quad$ status, result $\leftarrow$ UNCERTAINTYCHECK$(\rho, \mathcal{C}, \tau)$
8: **end while**
9: $z_q \leftarrow$ NONE, $\forall z \in \mathcal{C}$ $\quad \triangleright$ Policy iteration starts $(*)$
10: **for** $k \in 1, \ldots, K$ **do**
11: $\quad$ **for** $z \in \mathcal{C}$ **do**
12: $\quad\quad$ status, result $\leftarrow$ ROLLOUT$(n, H, \pi_{k-1}, z, \mathcal{C}, \tau)$
13: $\quad\quad$ **if** status $=$ DONE, **then** $z_q =$ result
14: $\quad\quad$ **else** $\mathcal{C} \leftarrow \mathcal{C} \cup \{\text{result}\}$ and **goto** line $(*)$
15: $\quad$ **end for**
16: $\quad w_k \leftarrow (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top q_{\mathcal{C}}$
17: $\quad \pi_k(a|s) \leftarrow \mathbb{1}\big(a = \arg\max_{\tilde{a} \in \mathcal{A}} w_k^\top \phi(s, \tilde{a})\big)$
18: **end for**
19: **return** π_{K-1}

gorithm until it stops returning a status of UNCERTAIN, and add the uncertain tuple to the core set each time. This is to ensure that the final policy[1] π_{K-1} returned by the main algorithm is approximately optimal from the initial state ρ, and this can be insured if all the uncertain actions (from ρ) are added to the core set (details in Appendix C).

In each iteration k, a Monte-Carlo estimation procedure (ROLLOUT, Algorithm 2) is launched for every element $z \in \mathcal{C}$ in the core set. An estimate (result in line 14) is obtained via taking the average return of n Monte-Carlo rollouts of length H while following policy π_{k-1}. ROLLOUT is *successful* if it returns a status of DONE and an estimate of $Q_{\pi_{k-1}}(z_s, z_a)$, which is assigned to z_q. If at iteration k ROLLOUT is successful for every core set element then z_q has a value estimate for all $z \in \mathcal{C}$, and the iteration is completed with the policy improvement step. The way the core set is constructed guarantees that the features of all the elements in the core set are sufficiently different to provide good target values $q_{\mathcal{C}}$ for least squares (Propositions 13 and 15).

Each time when ROLLOUT is *unsuccessful*, it returns a status of UNCERTAIN and a corresponding tuple. The uncertain tuple is added to the core set and policy iteration is restarted (line 14) and the value estimates for all the core set elements are reset to NONE (line 9). Roughly speaking, a tuple is flagged as uncertain when during the rollout

[1]The algorithm returns π_{K-1} instead of π_K because the proof requires that the uncertainty checks for the final policy pass. This is only ensured for π_{K-1}.

Algorithm 2 ROLLOUT

1: **Input:** number of rollouts n, length of rollouts H, rollout policy π, core set element z, core set $\mathcal{C}$, threshold τ.
2: **for** $i = 1, ..., n$ **do**
3: $\quad s_{i,0} \leftarrow z_s, a_{i,0} \leftarrow z_a$
4: $\quad$ Query the simulator, obtain $r_{i,0} \leftarrow r(s_{i,0}, a_{i,0})$, and the next state $s_{i,1}$
5: $\quad$ **for** $t = 1, ..., H$ **do**
6: $\quad\quad$ status, result$\leftarrow$UNCERTAINTYCHECK$(s_{i,t}, \mathcal{C}, \tau)$
7: $\quad\quad$ **if** status = UNCERTAIN **then**
8: $\quad\quad\quad$ **return** status, result
9: $\quad\quad$ **end if**
10: $\quad\quad$ Sample $a_{i,t} \sim \pi(\cdot|s_{i,t})$
11: $\quad\quad$ Query the simulator with $s_{i,t}, a_{i,t}$, obtain $r_{i,t} \leftarrow r(s_{i,t}, a_{i,t})$, and next state $s_{i,t+1}$
12: $\quad$ **end for**
13: **end for**
14: result $\leftarrow \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{H} \gamma^t r_{i,t}$
15: **return** DONE, result

a features is observed that is sufficiently different from all the features in the core set $\{z_\phi : z \in \mathcal{C}\}$. Important is that adding tuples to the core set in this way ensures that the size of the core set is bounded by a $\mathcal{O}(d)$ (Lemma 7). Restarting policy iteration is mainly to simplify the analysis; in practice it is reasonable to continue with the same policy.

It remains to specify the UNCERTAINTYCHECK subroutine that is used in Algorithms 1 and 2. For a fixed state $s \in \mathcal{S}$ the purpose of the uncertainty check is to search for an *uncertain action* that satisfies

$$\phi(s, a)^\top (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \phi(s, a) > \tau \qquad (1)$$

Here $\Phi_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}| \times d}$ is a matrix of all the features from the tuples in the core set stacked vertically. Solving Eq. (1) *exactly* recovers the approach by Yin et al. (2021). However, as this amounts to solving a positive-definite maximization problem, this is infeasible in general.

4.1 Efficient Good Set Search (EGSS, Algorithm 3)

Next, we show how to efficiently approximate the uncertainty check in Eq. (1). Define $V_{\mathcal{C}} = \Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I$ and a weighted matrix norm as $\|x\|_B^2 = x^\top B x$, $x \in \mathbb{R}^d, B \in \mathbb{R}^{d \times d}$. Using this notation, Eq. (1) becomes

$$\phi(s, a)^\top (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \phi(s, a) = \|\phi(s, a)\|_{V_{\mathcal{C}}^{-1}}^2 > \tau.$$

We define the *good set* to be the set of all features with $\|.\|_{V_{\mathcal{C}}^{-1}}^2$ weighted norm less than or equal to τ as follows

$$\mathcal{D} = \{\phi(s, a) : \|\phi(s, a)\|_{V_{\mathcal{C}}^{-1}}^2 \leq \tau\}.$$

Fix a state $s \in \mathcal{S}$. We want to check if there exists an action outside of the good set (i.e. $a \in \mathcal{A}$

Algorithm 3 UNCERTAINTYCHECK-EGSS

Input: state s, core set $\mathcal{C}$, threshold τ
$L \leftarrow \text{Cholesky}((\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1})$
for $v \in \{\pm e_l\}_{l=1}^d$ **do**
  $\hat{a} \leftarrow \arg\max_{a \in \mathcal{A}} \phi(s,a)^\top Lv$
  if $\left(\phi(s,\hat{a})^\top Lv\right)^2 > \tau$ **then**
    result $\leftarrow (s, \hat{a}, \phi(s,\hat{a}), \text{NONE})$
    return UNCERTAIN, result
  end if
end for
return CERTAIN, NONE

that satisfies $\|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} > \tau$) with computation that does not depend on $|\mathcal{A}|$. To this end, let $LL^\top = V_{\mathcal{C}}^{-1}$ be a Cholesky decomposition of $V_{\mathcal{C}}^{-1}$ and define $\hat{a} = \arg\max_{a \in \mathcal{A}} \|L^\top \phi(s,a)\|_\infty$. Note that $\hat{a}$ satisfies the following norm inequalities:

$$\frac{1}{d}\|\phi(s,\hat{a})\|^2_{V_{\mathcal{C}}^{-1}} \le \|L^\top \phi(s,\hat{a})\|^2_\infty \le \|\phi(s,\hat{a})\|^2_{V_{\mathcal{C}}^{-1}}$$

In other words, if $\|L^\top \phi(s,\hat{a})\|^2_\infty > \tau$ holds, then we have $\|\phi(s,\hat{a})\|^2_{V_{\mathcal{C}}^{-1}} > \tau$ and we have found an uncertain state-action pair. At the same time if $\|L^\top \phi(s,\hat{a})\|^2_\infty \le \tau$ then we are sure that $\|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \le d\tau$ for all $a \in \mathcal{A}$. The fact that the last inequality is still sufficient to provide bounds on the sub-optimality of policy evaluation manifests in Proposition 13, where only an extra factor of $\sqrt{d}$ is introduced. Finally, notice that

$$\max_{a \in \mathcal{A}} \|L^\top \phi(s,a)\|_\infty = \max_{v \in \{\pm e_i\}_{i=1}^d} \max_{a \in \mathcal{A}} \langle Lv, \phi(s,a)\rangle \quad (2)$$

can be computed efficiently using $2d$ calls to the greedy oracle (Assumption 3).

4.2 Theoretical Guarantees

The result that characterizes the performance of CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-EGSS is summarized in the next theorem.

Theorem 1 (CONFIDENT MC-LSPI EGSS Sub-Optimality)**.** *Suppose Assumptions 1 to 3 hold. If $\epsilon = 0$, for any $\kappa > 0$, with probability at least $1-\delta$, the final policy π_{K-1}, returned by* CONFIDENT MC-LSPI *combined with* UNCERTAINTYCHECK-EGSS *satisfies*

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \le \kappa.$$

The query and computation complexity are $\mathcal{O}\big(\frac{d^4}{\kappa^2(1-\gamma)^8}\big)$ and poly$(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ respectively. If $\epsilon > 0$, then with probability at least $1-\delta$, the policy π_{K-1}, output satisfies

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \le \tfrac{64\epsilon d}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

The query and computation complexity are $\mathcal{O}\big(\frac{d^2}{\epsilon^2(1-\gamma)^4}\big)$ and poly$(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$, respectively. All parameter settings are in Appendix E.

When compared to the result in Yin et al. (2021, Theorem 5.1) we have an extra factor of d in the query complexity for $\epsilon = 0$, while for $\epsilon \neq 0$ we only have an extra factor of $\sqrt{d}$ in the sub-optimality of the output policy. This is similar to linear bandits, where an extra $\sqrt{d}$ is suffered in the regret for oracle-efficient methods (Dani et al., 2008; Agrawal and Goyal, 2013; Abeille and Lazaric, 2017).

The full proof is given in Appendix E. The proof essentially follows the ideas in Yin et al. (2021) while carefully arguing how UNCERTAINTYCHECK-EGSS affects the query complexity. For the computational complexity, note that UNCERTAINTYCHECK-EGSS can be implemented in poly(d) by Eq. (2), and linear algebra operations. Since the core set size is bounded (Lemma 7), policy iteration only restarts $\mathcal{O}(d)$ times. Lastly, the policy improvement step is trivially implemented using the greedy oracle (Assumption 3).

5 ADDITIVE Q-FUNCTIONS

The result in Section 4.2 makes no restriction on the choice of features as long as the greedy policy can be computed efficiently (Assumptions 1 and 3). Next, we introduce an additive feature model for which the oracle can be implemented efficiently.

With the greedy oracle (Assumption 3), one can use CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-EGSS and directly invoke Theorem 1. However, in Section 5.1 we introduce a new uncertainty check algorithm, UNCERTAINTYCHECK-DAV, that explicitly uses the additive structure. The additive feature structure leads to improved results in the regimes where the dimension is large, but more importantly facilitates an efficient kernelized version of the CONFIDENT MC-LSPI algorithm (Section 5.2). The additive model also allows an efficient implementation of CONFIDENT MC-POLITEX (Yin et al., 2021), which leads to an improved dependence on the suboptimality in the misspecified setting (Section 5.3).

In the following, we assume that the action space can be decomposed into a product $\mathcal{A} = \mathcal{A}^{(1:m)} := \mathcal{A}^{(1)} \times \cdots \times \mathcal{A}^{(m)}$ for $m \ge 1$ (borrowing the standard notation from the multi-agent setting). We further assume access to feature maps $\phi_i : \mathcal{S} \times \mathcal{A}^{(i)} \to \mathbb{R}^d$ for each $i \in [m]$ and define $\phi(s, a^{(1:m)}) = \sum_{i=1}^m \phi_i(s, a^{(i)})$. The next assumption states that for any policy π, the Q_π-function is (approximately) linear in the feature map ϕ and decomposes additively across the components $\mathcal{A}^{(i)}$.

Assumption 4. *For each policy π there exists a weight vector $w_\pi \in \mathbb{R}^d, \|w_\pi\|_2 \le b$ satisfying*

$\max_{(s,a^{(1:m)})\in\mathcal{S}\times\mathcal{A}} |Q_\pi(s,a^{(1:m)}) - w_\pi^\top \sum_{j=1}^m \phi_j(s,a^{(j)})| \le \epsilon.$

In the context of the multi-agent setting (Example 1), the interpretation is that each $\phi_i(s,a^{(i)})$ models the contribution to the Q-function of each agent individually. Moreover, when Assumption 4 is satisfied, then for any weight vector $w \in \mathbb{R}^d$ the greedy policy can be implemented with $\mathcal{O}(d\sum_{i=1}^m |\mathcal{A}^{(i)}|)$ computation:

$$\begin{aligned}&\arg\max_{a^{(1:m)}\in\mathcal{A}} w^\top \phi(s,a^{(1:m)})\\ &= \big(\arg\max_{a^{(1)}\in\mathcal{A}^{(1)}} w^\top\phi_1(s,a^{(1)}), ..., \arg\max_{a^{(m)}\in\mathcal{A}^{(m)}} w^\top\phi_m(s,a^{(m)})\big)\end{aligned}$$

A simple example when Assumption 4 holds is when m agents "live" in m separate MDPs such that in each MDP the action-value functions are linearly realizable with their respective feature-maps and the goal is to maximize the sum of the rewards across the MDPs. In cases like this, we say that the "large" MDP is a *product MDP*. Note that in this setting agents only observe a joint reward after taking their actions, so an optimal policy for the joint MDP may not always be learned by simply applying single agent algorithms in each individual MDP. In Appendix F we show that Assumption 4 also captures MDPs that require cooperation between agents, and provide some empirical results.

5.1 Uncertainty Check using a Default Action Vector

In this section we introduce the uncertainty check with a default action vector (UNCERTAINTYCHECK-DAV, Algorithm 4). The goal of the uncertainty check is to ultimately bound the estimation error of w_k, i.e.

$$|w_k^\top\phi(s,a^{(1:m)}) - Q_{\pi_{k-1}}(s,a^{(1:m)})| \le \eta\,. \qquad (3)$$

Lemma 8 shows that a sufficient condition is to ensure that $\|\phi(s,a^{(1:m)})\|^2_{V_\mathcal{C}^{-1}} \le \tau$ for all $(s,a^{(1:m)}) \in (\mathcal{S}\times\mathcal{A})$ that are queried during policy evaluation.

We show that under Assumption 4, it is possible to achieve Eq. (3) while running the uncertainty check for a much smaller set of actions of size $\sum_{i=1}^m |\mathcal{A}^{(i)}|$. Recall that CONFIDENT MC-LSPI sets a *default action vector* $\bar{a}^{(1:m)} \in \mathcal{A}$ as a global. Define a subset of $\mathcal{A}$ as $\bar{\mathcal{A}}^{(1:m)} = \{(a^{(i)},\bar{a}^{(-i)}) : a^{(i)} \in \mathcal{A}^{(i)}, i \in [m]\}$, where we define $(a^{(i)},\bar{a}^{(-i)}) = (\bar{a}^{(1)},...,\bar{a}^{(i-1)},a^{(i)},\bar{a}^{(i+1)},...,\bar{a}^{(m)})$ as the action vector resulting from changing agent i's default action in $\bar{a}^{(1:m)}$ with $a^{(i)}$. Then, by Assumption 4 for any $a^{(1:m)} \in \mathcal{A}$ we have

$$\begin{aligned}&w_k^\top\phi(s,a^{(1:m)}) = w_k^\top\sum_{i=1}^m \phi_i(s,a^{(i)})\\ &= w_k^\top\Big(\sum_{i=1}^m \phi_i(s,a^{(i)}) \pm (m-1)\phi_i(s,\bar{a}^{(1:m)})\Big)\\ &= w_k^\top\Big(\sum_{i=1}^m \phi_i(s,(a^{(i)},\bar{a}^{(-i)})) - (m-1)\phi_i(s,\bar{a}^{(1:m)})\Big)\end{aligned}$$

Notice that $\bar{a}^{(1:m)}, (a^{(i)},\bar{a}^{(-i)}) \in \bar{\mathcal{A}}^{(1:m)}, \forall i \in [m]$. Thus, when $\|\phi(s,\tilde{a}^{(1:m)})\|^2_{V_\mathcal{C}^{-1}} \le \tau$ for all $\tilde{a}^{(1:m)} \in \bar{\mathcal{A}}^{(1:m)}$ we can ensure that for all action-vectors $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ that $|w_k^\top\phi(s,a^{(1:m)}) - Q_{\pi_{k-1}}(s,a^{(1:m)})| \le (2m-1)\eta$. In words, by checking the uncertainty of action-vectors that differ from the default action vector by at most one position $\tilde{a}^{(1:m)} \in \bar{\mathcal{A}}^{(1:m)}$ we can bound the sub-optimality of our estimate w_k, since the feature of any action vector can be related to the feature of the default action vector under Assumption 4. Since $\bar{\mathcal{A}}^{(1:m)}$ only contains $\sum_{i=1}^m |\mathcal{A}^{(i)}|$ elements, this procedure is poly$(d, \sum_{i=1}^m |\mathcal{A}^{(i)}|)$.

Algorithm 4 UNCERTAINTYCHECK-DAV

Input: state s, core set $\mathcal{C}$, threshold τ.
Globals: number of action components m
for $j \in [m]$ **do**
  for $a^{(j)} \in \mathcal{A}^{(j)}$ **do**
    $\tilde{a} \leftarrow (a^{(j)}, \bar{a}^{(-j)})$
    if $\phi(s,\tilde{a})^\top V_\mathcal{C}^{-1}\phi(s,\tilde{a}) > \tau$ **then**
      result $\leftarrow (s,\tilde{a},\phi(s,\tilde{a}),$ NONE$)$
      return UNCERTAIN, result
    end if
  end for
end for
return CERTAIN, NONE

The result that characterizes the performance of CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-DAV is summarized in the next theorem.

Theorem 2 (CONFIDENT MC-LSPI DAV Sub-Optimality)**.** *Suppose Assumption 4, and 2 hold. If $\epsilon = 0$, for any $\kappa > 0$, with probability at least $1-\delta$, the policy π_{K-1}, output by* CONFIDENT MC-LSPI *combined with* UNCERTAINTYCHECK-DAV *satisfies*

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \le \kappa.$$

The query and computation complexity are $\mathcal{O}\left(\frac{m^2d^3}{\kappa^2(1-\gamma)^8}\right)$. and poly$(\sum_{i=1}^m|\mathcal{A}^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ respectively. If $\epsilon > 0$, then with probability at least $1-\delta$, the output policy π_{K-1} satisfies

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \le \frac{128\epsilon\sqrt{dm}}{(1-\gamma)^2}(1+\log(1+b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

The query and computation complexity are $\mathcal{O}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$. and poly$(\sum_{i=1}^m|\mathcal{A}^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ respectively. All parameter settings are in Appendix E.

When compared to the result in Yin et al. (2021, Theorem 5.1) we have an extra factor of m^2 in the query complexity for $\epsilon = 0$, while for $\epsilon \neq 0$ we only have an extra factor of m in the sub-optimality of the output policy. On the other hand, the computational complexity is improved from $\mathcal{O}(\prod_{i=1}^m |\mathcal{A}^{(i)}|)$ for the prior

work to $\mathcal{O}(\sum_{i=1}^{m}|\mathcal{A}^{(i)}|)$. When compared to Theorem 1 where UNCERTAINTYCHECK-EGSS was used instead of UNCERTAINTYCHECK-DAV the extra dependence on $\sqrt{d}$ changed to m.

5.2 Kernelized Setting

The kernelized setting is a standard extension of the finite-dimensional linear setup (Srinivas et al., 2009; Abbasi-Yadkori, 2012). It lifts the restriction that the features and parameter vector are elements of $\mathbb{R}^d$. Formally the kernel is $\mathbf{k} : (\mathcal{S} \times \mathcal{A}^{(1:m)})^2 \to \mathbb{R}$, which gives rise to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, defined as a vector space $V_{\mathcal{H}} := \mathbb{R}^{\mathcal{S}\times\mathcal{A}^{(1:m)}}$ with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}} : V_{\mathcal{H}} \times V_{\mathcal{H}} \to \mathbb{R}$. We require that the Q_π-function is approximately contained in an RKHS. This includes cases where the linear dimension of function class is infinite.

Assumption 5 (Kernel Q_π-realizability)**.** *For each policy π there exists a vector $\tilde{Q}_\pi \in \mathcal{H}, \|\tilde{Q}_\pi\|_{\mathcal{H}} \leq b$ that satisfies $\sup_{s\in\mathcal{S},a^{(1:m)}\in\mathcal{A}^{(1:m)}} |Q_\pi(s,a^{(1:m)}) - \tilde{Q}_\pi(s,a^{(1:m)})| \leq \epsilon$, where $\tilde{Q}_\pi(s,a^{(1:m)}) = \langle\tilde{Q}_\pi, \mathbf{k}(s,a^{(1:m)},\cdot,\cdot)\rangle_{\mathcal{H}}$.*

Similar to the finite setting we assume an additive structure (on the kernel now) to allow efficient implementation. For component $j \in [m]$, define the kernel as $\mathbf{k}_j : (\mathcal{S} \times \mathcal{A}^{(j)})^2 \to \mathbb{R}$, which gives rise to an RKHS $\mathcal{H}_j$, defined as a vector space $V_{\mathcal{H}_j} := \mathbb{R}^{\mathcal{S}\times\mathcal{A}^{(j)}}$ with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}} : V_{\mathcal{H}_j} \times V_{\mathcal{H}_j} \to \mathbb{R}$.

Assumption 6. *The kernel $\mathbf{k}$ can be written as $\mathbf{k}(s_1,a_1^{(1:m)},s_2,a_2^{(1:m)}) = \sum_{j=1}^{m}\mathbf{k}_j(s_1,a_1^{(j)},s_2,a_2^{(j)})$ where $s_1,s_2 \in \mathcal{S},\ a_1^{(1:m)},a_2^{(1:m)} \in \mathcal{A}^{(1:m)}$.*

The kernel setting requires us to address two main challenges. First, the scaling of the query complexity with the dimension d needs to be improved to a notion of effective dimension. Following Du et al. (2021); Huang et al. (2021b) we make use of the critical information gain $\tilde{\Gamma}$ (defined in Eq. (25), Appendix D) which can be bounded for different RKHS of interest (Srinivas et al., 2009; Huang et al., 2021b). Second, computationally we cannot directly work with infinite dimensional features $\phi(s,a) = \mathbf{k}(s,a,\cdot,\cdot)$. Instead, we rely on the 'kernel trick' and compute all quantities of interest in the finite-dimensional data space (Schölkopf et al., 2001). After formally arguing as stated above, one can show that a kernelized version of CONFIDENT MC-LSPI and UNCERTAINTYCHECK-DAV provide the following sub-optimality guarantees on the output policy (proof in Appendix D).

Theorem 3 (CONFIDENT KERNEL MC-LSPI DAV Sub--Optimality)**.** *Suppose Assumption 5, 6, and 2 hold. Define $\tilde{\Gamma} := \tilde{\Gamma}(\lambda,\log(2))$. If $\epsilon = 0$, for any $\kappa > 0$, with probability at least $1-\delta$, the policy π_{K-1}, returned by* CONFIDENT KERNEL MC-LSPI *(Algorithm 7) combined with* UNCERTAINTYCHECK-K-DAV *(Algorithm 8) satisfies*

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \kappa.$$

The query and computation complexity are $\mathcal{O}\left(\frac{m^2\tilde{\Gamma}^3}{\kappa^2(1-\gamma)^8}\right)$ and poly$(\sum_{i=1}^{m}|\mathcal{A}^{(i)}|,\tilde{\Gamma},\frac{1}{1-\gamma},\frac{1}{\kappa},\log(\frac{1}{\delta}))$ respectively. If $\epsilon > 0$, then with probability at least $1-\delta$, the final policy π_{K-1} satisfies

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \frac{32\epsilon m\sqrt{\tilde{\Gamma}}}{(1-\gamma)^2}.$$

The query and computation complexity are $\mathcal{O}\left(\frac{\tilde{\Gamma}^2}{\epsilon^2(1-\gamma)^4}\right)$ and poly$(\sum_{i=1}^{m}|\mathcal{A}^{(i)}|,\tilde{\Gamma},\frac{1}{1-\gamma},\frac{1}{\epsilon},\log(\frac{1}{\delta}),\log(1+b))$ respectively. All parameter settings are in Appendix E.

The result is identical to Theorem 2 except with d replaced with the critical information gain $\tilde{\Gamma}(\lambda,\log(2))$.

5.3 Politex

The Politex algorithm has been shown to obtain better sub-optimality gaurantees than LSPI by Abbasi-Yadkori et al. (2019). In this section we show that CONFIDENT MC-POLITEX presented by Yin et al. (2021) can be extended to combinatorially large action spaces. Although Politex is a also based on policy iteration, like LSPI, an important difference is that it uses stochastic policies based on an exponential weighting of each actions Q-value. Efficiently sampling from such a policy is not always possible when the action space is combinatorially large. We show Assumption 4 is sufficient to do so (Proposition 6). Moreover, using similar arguments as in Section 5.1, indeed, CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-DAV achieves better sub-optimality guarantees than CONFIDENT MC-LSPI.

Theorem 4 (CONFIDENT MC-POLITEX Sub-Optimality)**.** *Suppose Assumption 4, and 2 hold. If $\epsilon > 0$, for any $\kappa > 0$, with probability at least $1-\delta$, the policy $\bar{\pi}_{K-1}$, output by* CONFIDENT MC-POLITEX *(Algorithm 6) combined with* UNCERTAINTYCHECK-DAV *(Algorithm 4) satisfies*

$$V^*(\rho) - V_{\bar{\pi}_{K-1}}(\rho) \leq \frac{64\epsilon m\sqrt{d}}{1-\gamma}(1+\log(1+b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

The query and computation complexity are $\mathcal{O}\left(\frac{md}{\epsilon^4(1-\gamma)^5}\right)$ and poly$(\sum_{i=1}^{m}|\mathcal{A}^{(i)}|,d,\frac{1}{1-\gamma},\frac{1}{\epsilon},\log(\frac{1}{\delta}),\log(1+b))$ respectively. All parameter settings are in Appendix E.

As expected the sub-optimality is better (scales with $1/(1-\gamma)$) than that of CONFIDENT MC-LSPI(Theorem 2), which scales with $1/(1-\gamma)^2$. However, the query complexity is worse (as is typical for Politex), and an extra factor of m is introduced, since mirror descent needs to be run on the entire action space of size $\prod_{i=1}^{m}|\mathcal{A}^{(i)}|$ for each state. We also extend the result to the kernelized setting in Appendix E, and show that UNCERTAINTYCHECK-EGSS can be used when Assumption 4 is satisfied.

6 CONCLUSION

In this work, we considered the problem of planning with a local access simulator when the action space is combinatorially large. We introduced several algorithms that achieve polynomial computational and query complexity guarantees, while still maintaining a reasonable sub-optimality of the output policy under various assumptions. The main novelty is an efficient implementation of the uncertainty check under the mild assumption of having access to a greedy oracle. If the Q-functions for all policies satisfy an additive structure we provide nuanced results that show how the sample complexity can be improved in the regime where the dimension is large. Under the same additive structure our results also extend to the kernelized setting. An interesting direction for future work is to extend the results to the Factored MDP model (Guestrin et al., 2001) or the Confident LSVI algorithm (Hao et al., 2022).

Acknowledgements

Johannes Kirschner gratefully acknowledges funding from the SNSF Early Postdoc.Mobility fellowship P2EZP2_199781. Matej Jusup gratefully acknowledges support by the Swiss National Science Foundation under the research project DADA/181210. Csaba Szepesvári gratefully acknowledges funding from the Canada CIFAR AI Chairs Program, Amii and NSERC.

References

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602 (7897):414–419, 2022.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Dong Yin, Botao Hao, Yasin Abbasi-Yadkori, Nevena Lazić, and Csaba Szepesvári. Efficient local planning with linear function approximation. *arXiv preprint arXiv:2108.05533*, 2021.

Dimitri P Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. *Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA*, 14, 1996.

Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

Rémi Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1006. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pages 5662–5670. PMLR, 2020.

Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in mdps with linearly-realizable optimal action-value functions. In *Algorithmic Learning Theory*, pages 1237–1264. PMLR, 2021.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019.

Dong Chen, Kaian Chen, Zhaojian Li, Tianshu Chu, Rui Yao, Feng Qiu, and Kaixiang Lin. Powernet: Multi-agent deep reinforcement learning for scalable power-grid control. *IEEE Transactions on Power Systems*, 37 (2):1007–1017, 2021a.

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

Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Ronald A Howard. Dynamic programming and markov processes. 1960.

Rémi Munos. Error bounds for approximate policy iteration. In *ICML*, volume 3, pages 560–567, 2003.

Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. Error propagation for approximate policy and value iteration. *Advances in Neural Information Processing Systems*, 23, 2010.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? *arXiv preprint arXiv:1910.03016*, 2019.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *URL https://rltheorybook. github. io*, 2020a.

Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellért Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pages 3692–3702. PMLR, 2019.

Csaba Szepesvári. Lecture notes in reinforcement learning theory, Aug 2022a. URL `https://rltheory.github.io/lecture-notes/planning-in-mdps/lec13/`.

Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1):33–57, 1996.

Francisco S Melo and M Isabel Ribeiro. Q-learning with linear function approximation. In *International Conference on Computational Learning Theory*, pages 308–322. Springer, 2007.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pages 2377–2386. PMLR, 2016.

Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I Jordan. On function approximation in reinforcement learning: optimism in the face of large state spaces. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 13903–13916, 2020.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning*, pages 10978–10989. PMLR, 2020.

Simon Du, Sham Kakade, Jason Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear classes: A structural framework for provable generalization in rl. In *International Conference on Machine Learning*, pages 2826–2836. PMLR, 2021.

Dongruo Zhou, Quanquan Gu, and Csaba Szepesvari. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*, pages 4532–4576. PMLR, 2021.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *Advances in neural information processing systems*, 33:13399–13412, 2020b.

Nevena Lazic, Dong Yin, Yasin Abbasi-Yadkori, and Csaba Szepesvari. Improved regret bound and experience replay in regularized policy iteration. In *International Conference on Machine Learning*, pages 6032–6042. PMLR, 2021.

Chen-Yu Wei, Mehdi Jafarnia Jahromi, Haipeng Luo, and Rahul Jain. Learning infinite-horizon average-reward mdps with linear function approximation. In *International Conference on Artificial Intelligence and Statistics*, pages 3007–3015. PMLR, 2021.

Botao Hao, Nevena Lazic, Dong Yin, Yasin Abbasi-Yadkori, and Csaba Szepesvari. Confident least square value iteration with local access to a simulator. In *International Conference on Artificial Intelligence and Statistics*, pages 2420–2435. PMLR, 2022.

Thomas L Dean, Robert Givan, and Kee-Eung Kim. Solving stochastic planning problems with large state and action spaces. In *AIPS*, pages 102–110, 1998.

Florian Geißer, David Speck, and Thomas Keller. Trial-based heuristic tree search for mdps with factored action spaces. In *Thirteenth Annual Symposium on Combinatorial Search*, 2020.

Aswin Raghavan, Saket Joshi, Alan Fern, Prasad Tadepalli, and Roni Khardon. Planning in factored action spaces with symbolic dynamic programming. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. *Advances in Neural Information Processing Systems*, 27, 2014.

Ziping Xu and Ambuj Tewari. Near-optimal reinforcement learning in factored mdps: Oracle-efficient algorithms for the non-episodic setting. *Advances in Neural Information Processing Systems*, 33, 2020.

Yi Tian, Jian Qian, and Suvrit Sra. Towards minimax optimal reinforcement learning in factored markov decision processes. *Advances in Neural Information Processing Systems*, 33:19896–19907, 2020.

Xiaoyu Chen, Jiachen Hu, Lihong Li, and Liwei Wang. Efficient reinforcement learning in factored mdps with application to constrained rl. *arXiv preprint arXiv:2008.13319*, 2020.

Arthur Delarue, Ross Anderson, and Christian Tjandraatmadja. Reinforcement learning with combinatorial actions: An application to vehicle routing. *Advances in Neural Information Processing Systems*, 33:609–620, 2020.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pages 4476–4486. PMLR, 2021.

Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78 (5):1404–1422, 2012.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International conference on machine learning*, pages 151–159. PMLR, 2013.

Alexander Shleyfman, Antonín Komenda, and Carmel Domshlak. On combinatorial actions and cmabs with linear side information. In *ECAI 2014*, pages 825–830. IOS Press, 2014.

Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, et al. Combinatorial bandits revisited. *Advances in neural information processing systems*, 28, 2015.

Marc Jourdan, Mojmír Mutnỳ, Johannes Kirschner, and Andreas Krause. Efficient pure exploration for combinatorial bandits with semi-bandit feedback. In *Algorithmic Learning Theory*, pages 805–849. PMLR, 2021.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pages 295–304. PMLR, 2015.

Kai Wang, Bryan Wilder, Sze-chuan Suen, Bistra Dilkina, and Milind Tambe. Improving gp-ucb algorithm by harnessing decomposed feedback. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 555–569. Springer, 2019.

Johannes Kirschner and Andreas Krause. Bias-robust bayesian optimization via dueling bandits. In *International Conference on Machine Learning*, pages 5595–5605. PMLR, 2021.

Mojmir Mutny and Andreas Krause. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, 31, 2018.

Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics*, pages 298–307. PMLR, 2018.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2): 156–172, 2008.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multiagent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

Ziang Song, Song Mei, and Yu Bai. When can we learn general-sum markov games with a large number of players sample-efficiently? *arXiv preprint arXiv:2110.04184*, 2021.

Yi Tian, Yuanhao Wang, Tiancheng Yu, and Suvrit Sra. Online learning in unknown markov games. In *International conference on machine learning*, pages 10279–10288. PMLR, 2021.

Yu Bai and Chi Jin. Provable self-play algorithms for competitive reinforcement learning. In *International conference on machine learning*, pages 551–560. PMLR, 2020.

Qinghua Liu, Tiancheng Yu, Yu Bai, and Chi Jin. A sharp analysis of model-based reinforcement learning with self-play. In *International Conference on Machine Learning*, pages 7001–7010. PMLR, 2021.

Stefanos Leonardos, Will Overman, Ioannis Panageas, and Georgios Piliouras. Global convergence of multiagent policy gradient in markov potential games. *arXiv preprint arXiv:2106.01969*, 2021.

Chi Jin, Qinghua Liu, Yuanhao Wang, and Tiancheng Yu. V-learning–a simple, efficient, decentralized algorithm for multiagent rl. *arXiv preprint arXiv:2110.14555*, 2021.

Baihe Huang, Jason D Lee, Zhaoran Wang, and Zhuoran Yang. Towards general function approximation in zero-sum markov games. *arXiv preprint arXiv:2107.14702*, 2021a.

Zixiang Chen, Dongruo Zhou, and Quanquan Gu. Almost optimal algorithms for two-player markov games

with linear function approximation. *arXiv preprint arXiv:2102.07404*, 2021b.

Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, pages 5571–5580. PMLR, 2018.

Barna Pasztor, Ilija Bogunovic, and Andreas Krause. Efficient model-based multi-agent mean-field reinforcement learning. *arXiv preprint arXiv:2107.04050*, 2021.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. *Advances in neural information processing systems*, 14, 2001.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

Roy Zohar, Shie Mannor, and Guy Tennenholtz. Locality matters: A scalable value decomposition approach for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2109.10632*, 2021.

Vijay Bhattiprolu, Euiwoong Lee, and Assaf Naor. A framework for quadratic form maximization over convex sets through nonconvex relaxations. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 870–881, 2021.

Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. 2008.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. In *Artificial Intelligence and Statistics*, pages 176–184. PMLR, 2017.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Yasin Abbasi-Yadkori. *Online Learning for Linearly Parametrized Control Problems*. PhD thesis, Citeseer, 2012.

Kaixuan Huang, Sham M Kakade, Jason D Lee, and Qi Lei. A short note on the relationship of information gain and eluder dimension. *arXiv preprint arXiv:2107.02377*, 2021b.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

Szepesvári. Politex, Aug 2022b. URL `https://rltheory.github.io/lecture-notes/planning-in-mdps/lec14/`.

Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 82–90. PMLR, 2021.

A EFFICIENT POLICY SAMPLING

The policy in (Yin et al., 2021) for CONFIDENT MC-LSPI and CONFIDENT MC-POLITEX is as follows

$$\pi_k(a|s) \leftarrow \begin{cases} \mathbb{1}\left(a = \arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})\right) & \text{LSPI} \\ \exp\left(\alpha \sum_{j=0}^{k-1} Q_j(s,a)\right) / \sum_{a\in\mathcal{A}} \exp\left(\alpha \sum_{j=0}^{k-1} Q_k(s,a)\right). & \text{Politex} \end{cases} \tag{4}$$

with $w_k = (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} \Phi_\mathcal{C}^\top q_\mathcal{C}$ and $Q_{k-1}(s,a) = \min\{\max\{w_k^\top \phi(s,a), 0\}, 1/(1-\gamma)\}$ for the Politex case only. In this section we show that the above policy can be sampled from efficiently if Assumption 4 or Assumption 3 is satisfied for the LSPI case and policy π_k can be sampled from efficiently if Assumption 4 is satisfied for the Politex case. To be precise, by efficiently we mean with computation that does not depend on $|\mathcal{A}|$. We assume only $w \in \mathbb{R}^d$ or $w_0, ..., w_{k-1} \in \mathbb{R}^d$ (for LSPI and Politex respectively) and a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ are given, thus the process of sampling may require calculating the policy if necessary to accurately sample. First we handle the LSPI case.

Proposition 5 (Efficient LSPI Policy Sampling)**.** *Given state $s \in \mathcal{S}$, parameter vector $w \in \mathbb{R}^d$, feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and assumption Assumption 4 or Assumption 3 satisfied. Then policy*

$$\pi_k(a|s) = \mathbb{1}\left(a = \arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})\right)$$

can be sampled from in with computation that does not depend on $|\mathcal{A}|$.

Proof. One can sample from policy π_k by simply outputting the result of $\arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})$. Under assumption Assumption 3 $\arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})$ can be computed in constant time by applying the oracle to w and ϕ (i.e. $\mathcal{G}(w,\phi)$). While, Assumption 4 implies we can compute $\arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})$ in poly$(\sum_{i=1}^m |A^{(i)}|, d)$ time, since

$$\begin{aligned} &\arg\max_{a^{(1:m)}\in\mathcal{A}^{(1:m)}} w^\top \phi(s, a^{(1:m)}) \\ &= \big(\arg\max_{a^{(1)}\in\mathcal{A}^{(1)}} w^\top \phi_1(s,a^{(1)}), ..., \arg\max_{a^{(m)}\in\mathcal{A}^{(m)}} w^\top \phi_m(s,a^{(m)})\big) \end{aligned}$$

□

Next, we handle the Politex case. To achieve the result below we assume Assumption 4 is satisfied. We have to modify the Politex policy in Eq. (4) slightly, by removing the clipping of the Q-function at each iteration k (i.e. we define the Q-function at iteration k to be $Q_{k-1}(s,a) = w_k^\top \phi(s,a)$ instead of $Q_{k-1}(s,a) = \min\{\max\{w_k^\top \phi(s,a), 0\}, 1/(1-\gamma)\}$). This was done since we were not aware of an efficient way to compute the clipped Q-function for all action-vectors in $\mathcal{A}^{(1:m)}$. Importantly, removing the clipping does not suffer any increase in the dominating terms of the final policies sub-optimality (shown in Appendix E)

Proposition 6 (Efficient Politex Policy Sampling)**.** *Given state $s \in \mathcal{S}$, parameter vectors $w_0, ..., w_{k-1} \in \mathbb{R}^d$, feature map $\phi : \mathcal{S} \times \mathcal{A}^{(1:m)} \to \mathbb{R}^d$ and Assumption 4 satisfied. Then policy*

$$\pi_k(a^{(1:m)}|s) = \exp\left(\alpha \sum_{j=0}^{k-1} w_j^\top \phi(s, a^{(1:m)})\right) / \sum_{\tilde{a}^{(1:m)}\in\mathcal{A}^{(1:m)}} \exp\left(\alpha \sum_{j=0}^{k-1} w_j^\top \phi(s, \tilde{a}^{(1:m)})\right)$$

with $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ can be sampled from in time poly$(\sum_{i=1}^m |\mathcal{A}_i|, d)$.

Proof. Fix arbitrary $a^{(1:m)} \in \mathcal{A}^{(1:m)}$. To sample from π_k it is sufficient to sample actions $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ proportional to $\exp(\alpha \sum_{j=0}^{k-1} Q_j(s, a^{(1:m)}))$. Rearranging $\exp(\alpha \sum_{j=0}^{k-1} Q_j(s, a^{(1:m)}))$ and plugging in that $\phi(s, a^{(1:m)}) =$

$\sum_{i=1}^{m} \phi_i(s, a^{(i)})$ under assumption Assumption 4 we have

$$\begin{aligned}\exp\left(\alpha \sum_{j=0}^{k-1} w_j^\top \phi(s, a^{(1:m)})\right) &= \prod_{j=0}^{k-1} \exp\left(\alpha w_j^\top \phi(s, a^{(1:m)})\right) \\ &= \prod_{j=0}^{k-1} \exp\left(\alpha w_j^\top \sum_{i=1}^{m} \phi_i(s, a^{(i)})\right) \\ &= \prod_{i=1}^{m} \prod_{j=0}^{k-1} \exp\left(\alpha w_j^\top \phi_i(s, a^{(i)})\right)\end{aligned}$$

Which means that the probability of sampling action $a^{(1:m)}$ is equal to the product of the probabilities of sampling $a^{(i)}$ for $i \in [m]$ independently. Since $a^{(1:m)}$ was arbitrary this completes the proof. □

B BOUND ON CORE SET SIZE

Yin et al. (2021) showed that when only tuples containing state-action vectors that satisfy $\phi(s,a)^\top(\Phi^\top\Phi+\lambda I)^{-1}\phi(s,a) > \tau$ are add to the core set then it can be bounded as follows.

Lemma 7 (Bound on Core Set Size (Lemme 5.1 in (Yin et al., 2021))). *When Assumption 2 is satisfied, and $(s,a) \in (\mathcal{S}\times\mathcal{A})$ that satisfy $\phi(s,a)^\top(\Phi_\mathcal{C}^\top\Phi_\mathcal{C} + \lambda I)^{-1}\phi(s,a) > \tau$ are added to the core set, the size of the core set can be bounded by*

$$\tilde{C}_{\max} := \frac{e}{e-1}\frac{1+\tau}{\tau} d\left(\log(1+\frac{1}{\tau}) + \log(1+\frac{1}{\lambda})\right). \tag{5}$$

C EFFICIENT UNCERTAINTY CHECK

The CONFIDENT MC-LSPI algorithm proposed by Yin et al. (2021) is the same as our CONFIDENT MC-LSPI (Algorithm 1) algorithm combined with UNCERTAINTYCHECK (Algorithm 5) and the policy on line 17 of CONFIDENT MC-LSPI replaced with Eq. (4). Notice that UNCERTAINTYCHECK requires iterating over $\mathcal{A}$ (line 2), which is computa-

Algorithm 5 UNCERTAINTYCHECK

1: **Input:** state s, core set $\mathcal{C}$, threshold τ
2: **for** $a \in \mathcal{A}$ **do**
3: **if** $\phi(s,a)^\top(\Phi_\mathcal{C}^\top\Phi_\mathcal{C} + \lambda I)^{-1}\phi(s,a) > \tau$ **then**
4: status $\leftarrow$ UNCERTAIN, result $\leftarrow (s, a, \phi(s,a), \text{NONE})$
5: **return** status, result
6: **end if**
7: **end for**
8: **return** CERTAIN, NONE

tionally expensive with the action space is combinatorially large. In this appendix we show how the loop over all actions $a \in \mathcal{A}$ in the UNCERTAINTYCHECK algorithm can be avoided when either Assumption 4 or Assumption 3 is satisfied. In particular, we show that UNCERTAINTYCHECK-DAV and UNCERTAINTYCHECK-EGSS algorithms are able to reduce the computation time of UNCERTAINTYCHECK to no longer depend on $|\mathcal{A}|$, while still maintaining suitable output policy guarantees.

Since, we are extending the CONFIDENT MC-LSPI algorithm proposed by Yin et al. (2021), we will be borrowing much of the steps from their proof. Yin et al. (2021) used a *virtual algorithm* (VA) and *main algorithm* (MA) to prove the sub-optimality of their CONFIDENT MC-LSPI algorithm. We give a brief summary of the VA and MA; however, avoid full details since we use the exact same definition as in Yin et al. (2021). Until the next subsection, assume UNCERTAINTY-CHECK is used in CONFIDENT MC-LSPI and ROLLOUT. The MA is exactly CONFIDENT MC-LSPI. The VA is based on the CONFIDENT MC-LSPI algorithm, but has some differences, which we outline next. The VA runs for exactly $C_{\max}$ loops, K iterations, and completes all n of its rollouts of length H. For each loop and iteration k the VA always obtains

estimates $q_{\mathcal{C}}$ of its policy. The VA uses a different policy than the MA for rollouts. We will first focus on the LSPI case and return to Politex much later. The VA's Q-function at iteration k is

$$\tilde{Q}_{k-1}(s,a) = \begin{cases} \tilde{w}_k^\top \phi(s,a) & \text{if } \phi(s,a) \in \mathcal{D} \\ Q_{\tilde{\pi}_{k-1}}(s,a) & \text{if } \phi(s,a) \notin \mathcal{D} \end{cases}$$

where $\tilde{w}_k = V_{\mathcal{C}}^{-1} \Phi_{\mathcal{C}}^\top \tilde{q}_{\mathcal{C}}$, and $\tilde{q}_{\mathcal{C}}$ are the estimates obtained from running ROLLOUT on each element of the core set, and $\mathcal{D} = \{\phi(s,a) : \|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \leq \tau\}$ is the *good set*. The VA's policy is

$$\tilde{\pi}_k(a|s) = \mathbb{1}\left(a = \arg\max_{\tilde{a} \in \mathcal{A}} \tilde{Q}_{k-1}(s,\tilde{a})\right).$$

The nice thing about defining the VA's policy in this way is that we can make use of the following Lemma from (Yin et al., 2021).

Lemma 8 (Lemma B.2 in (Yin et al., 2021))**.** *Suppose that Assumption Assumption 4 holds. With all terms as defined earlier and $\theta > 0$. Then, with probability at least*

$$1 - 2C_{max} \exp(-2\theta^2 (1-\gamma)^2 n)$$

for any $(s,a) \in (\mathcal{S} \times \mathcal{A})$ pair such that $\phi(s,a) \in \mathcal{D}$, we have

$$|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)| \leq b\sqrt{\lambda\tau} + \left(\epsilon + \frac{\gamma^{H-1}}{1-\gamma} + \theta\right)\sqrt{\tau C_{max}} + \epsilon := \eta$$

Notice that for any $(s,a) \in (\mathcal{S} \times \mathcal{A})$ pair such that $\phi(s,a) \notin \mathcal{D}$, the VA's Q-function $\tilde{Q}_{k-1}$ has access to the true Q-function $Q_{\tilde{\pi}_{k-1}}$ of policy $\tilde{\pi}_{k-1}$. Thus, we have that

$$\|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)\|_\infty \leq \eta \tag{6}$$

Combined with the fact that $\tilde{\pi}_k$ is greedy w.r.t. $\tilde{Q}_{k-1}$ the above result turns out to be especially useful.

To understand why, we state a classic policy improvement result, which can be found as Lemma B.3 in Yin et al. (2021) and in other papers.

Lemma 9 (approximate policy iteration)**.** *Suppose that we run K approximate policy iterations and generate a sequence of policies $\pi_0, \pi_1, \pi_2, \cdots, \pi_K$. Suppose that for every $k = 1, 2, \cdots, K$, in the k-th iteration, we obtain a function $\tilde{Q}_{k-1}$ such that, $\|\tilde{Q}_{k-1} - Q_{\pi_{k-1}}\|_\infty \leq \eta$, and choose π_k to be greedy with respect to $\tilde{Q}_{k-1}$. Then*

$$\|Q^* - Q_{\pi_K}\|_\infty \leq \frac{2\eta}{1-\gamma} + \frac{\gamma^K}{1-\gamma},$$

In our case the VA's policy $\tilde{\pi}_k$ is greedy w.r.t. $\tilde{Q}_{k-1}$ and thus we have that

$$\|Q^* - Q_{\tilde{\pi}_K}\|_\infty \leq \frac{2\eta}{1-\gamma} + \frac{\gamma^K}{1-\gamma},$$

Now we explain how the MA can be related to the VA, and make use of the above result. The UNCERTAINTYCHECK algorithm can have two cases:

Case 1: $\|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} > \tau$ holds for at least one $a \in \mathcal{A}$,

Case 2: $\|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \leq \tau$ holds for all $a \in \mathcal{A}$. This is equivalent to saying $\phi(s,a) \in \mathcal{D},\ \forall a \in \mathcal{A}$.

The VA is exactly the same at the MA algorithm, until Case 1 occurs for the first time. This is because the MA's and VA's simulators are coupled, in the sense that at iteration k, rollout i, and step t, when both simulators are queried with the same state-action vector pairs, they sample the exact same next state and reward. The VA also uses the same initial policy as the

MA at the start of policy iteration for every loop. Once Case 1 occurs the MA would restart policy iteration (else condition in line 14 of CONFIDENT MC-LSPI), while the VA does not. The VA records the state-action vector pair when Case 1 occurs for the first time and adds it to the core set once it completes running policy iteration for the current loop. In this way the core set maintained by the MA and VA are always the same. Since the size of the core set is bounded by $C_{\max}$ when $(s,a) \in (\mathcal{S} \times \mathcal{A})$ that satisfy $\phi(s,a)^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} \phi(s,a) > \tau$ are added to the core set (Lemma 7), there will be a loop of policy iteration at which the MA and VA never encounter Case 1 for any of the K iterations of policy iteration. We call this loop the *final loop*. This is equivalent to say that all $(s,a) \in (\mathcal{S} \times \mathcal{A})$ observed during all K iterations of policy iteration in the final loop are in the good set (i.e. $\phi(s,a) \in \mathcal{D}$). Notice that this means MA and VA behaved identical in the final loop, since the VA's policy would have always been greedy w.r.t. $\tilde{w}_k^\top \phi$ and the MA and VA use the same initial policy at the start of each loop. It turns out this relationship between the MA and VA allows us to bound the sub-optimality of the MA in the final loop, by using the result in Eq. (6) we have for the VA. More precisely, the following result can be extracted from (Yin et al., 2021)

Proposition 10 (equation (B.15) in Yin et al. (2021))**.** *With all terms as defined earlier. Define* $\eta \geq \|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)\|_\infty$. *Suppose* $\eta \geq |\tilde{w}_k^\top \phi(\rho,a) - Q_{\tilde{\pi}_{k-1}}(\rho,a)|,\ \forall a \in \mathcal{A}$. *Then, if the VA and MA behave identically in the final loop, with probability at least* $1 - 4KC_{max}^2 \exp(-2\theta^2(1-\gamma)^2 n)$ *we have*

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \frac{8\eta}{(1-\gamma)^2} + \frac{2\gamma^{K-1}}{(1-\gamma)^2} \tag{7}$$

Notice, that we require three things to use the above result. We need a bound on $\|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)\|_\infty$. We need a bound on $|\tilde{w}_k^\top \phi(\rho,a) - Q_{\tilde{\pi}_{k-1}}(\rho,a)|,\ \forall a \in \mathcal{A}$. We need to ensure that the VA and MA behave identically in the final loop. Then, we can get a bound on the sub-optimality of the MA's output policy π_{K-1}. An important observation is that UNCERTAINTYCHECK ensured that MA and VA behave identically in the final loop. It did this by making sure that the VA's policy $\tilde{\pi}_k$ would only be able to use $\tilde{w}_k^\top \phi$ to derive its actions, since UNCERTAINTYCHECK always returns a status of CERTAIN in the final loop, which means that $\phi(s,a) \in \mathcal{D}$ for all $s,a \in \mathcal{S} \times \mathcal{A}$ encountered in the final loop. With this information in mind, we now show that UNCERTAINTYCHECK-DAV and UNCERTAINTYCHECK-EGSS only requires computation independent of $|\mathcal{A}|$, while providing only slightly worse sub-optimality guarantees when compared to the result in (Yin et al., 2021).

C.1 Efficient Good Set Search Approach (EGSS)

In this section we prove some useful results for UNCERTAINTYCHECK-EGSS. Fix a state $s \in \mathcal{S}$. First, we show that with computation independent of $|\mathcal{A}|$, one can find an action vector $a \in \mathcal{A}$ that approximately maximizes $\phi(s,a)^\top V_\mathcal{C}^{-1} \phi(s,a)$.

Lemma 11 (Efficient good set search)**.** *Assume either Assumption 3 is satisfied. With all terms as defined earlier. One can ensure, with* $2d$ *calls to the greedy oracle that*

$$\phi(s,a)^\top V_\mathcal{C}^{-1} \phi(s,a) \leq d\tau$$

for all $a \in \mathcal{A}$*, or there exists an* $a \in \mathcal{A}$ *such that*

$$\phi(s,a)^\top V_\mathcal{C}^{-1} \phi(s,a) > \tau.$$

Further, if Assumption 4 is satisfied, then the same guarantees hold with $2d^2 \sum_{i=1}^m |A^{(i)}|$ *computation time.*

Proof. Recall that we are able to compute $\max_{a \in \mathcal{A}} \langle u, \phi(s,a) \rangle$ for any $u \in \mathbb{R}^d$ in constant time if Assumption 4 is satisfied, and in $d \sum_{i=1}^m |A^{(i)}|$ time if Assumption 3 is satisfied. We make use of a bi-directional 2-norm to ∞-norm inequality that will take advantage of the above mentioned efficient computation. Fix $\mathcal{C}$ and define the lower triangular matrix L via the Cholesky decomposition $V_\mathcal{C}^{-1} = LL^\top$. Define $\{e_i\}_{i=1}^d$ as the standard basis vectors and

$$(v^*, a_{\max}) := \arg \left(\max_{v \in \{\pm e_i\}_{i=1}^d} \max_{a \in \mathcal{A}} \langle Lv, \phi(s,a) \rangle \right)$$

Then we have that

$$\begin{aligned}
\frac{1}{d}\|\phi(s,a_{\max})\|^2_{V_{\mathcal{C}}^{-1}} &= \frac{1}{d}\phi(s,a_{\max})^\top V_{\mathcal{C}}^{-1}\phi(s,a_{\max}) \\
&= \frac{1}{d}\phi(s,a_{\max})^\top LL^\top\phi(s,a_{\max}) \\
&= \frac{1}{d}\|L^\top\phi(s,a_{\max})\|_2^2 \\
&\le \max_{a\in\mathcal{A}}\|L^\top\phi(s,a)\|_\infty^2 \\
&= \max_{v\in\{\pm e_i\}_{i=1}^d}\max_{a\in\mathcal{A}}\langle v, L^\top\phi(s,a)\rangle^2 \\
&= \max_{v\in\{\pm e_i\}_{i=1}^d}\max_{a\in\mathcal{A}}\langle Lv, \phi(s,a)\rangle^2 \qquad (8)\\
&= \langle Lv^*, \phi(s,a_{\max})\rangle^2 \\
&\le \|L^\top\phi(s,a_{\max})\|_2^2
\end{aligned}$$

The purpose of writing all the equalities up to Eq. (8) was to show that Eq. (8) can be computed efficiently. This is since we are able to compute $\max_{a\in\mathcal{A}}\langle Lv,\phi(s,a)\rangle^2$ in constant time if Assumption 4 is satisfied, and in $d\sum_{i=1}^m|A^{(i)}|$ time if Assumption 3 is satisfied, and $\{\pm e_i\}_{i=1}^d$ contains $2d$ elements. Also, note that L can be computed with at most d^2 computation in each loop by doing a rank one update to the Cholesky decomposition of $V_{\mathcal{C}}^{-1} = LL^\top$.

If equation (Eq. (8)) is larger than τ, then $\|\phi(s,a_{\max})\|^2_{V^{-1}} > \tau$. While, if equation (Eq. (8)) is less than or equal τ, then $\|\phi(s,a_{\max})\|^2_{V^{-1}} \le d\tau$, completing the proof. □

UNCERTAINTYCHECK-EGSS is essentially an implementation of equation (Eq. (8)), thus its computation is independent of $|\mathcal{A}|$, as stated in Lemma 11. Also, since only $a\in\mathcal{A}$ that satisfy $\|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \ge \|\phi(s,a)\|^2_\infty > \tau$ are added to the core set, we can still use Lemma 7 to bound the size of the core set by $C_{\max}$. Basically, Eq. (8) is an underestimate of $\|\phi(s,a_{\max})\|^2_{V^{-1}}$ and we only add elements to the core set when it is larger than τ, thus the core set is no larger than it was when using UNCERTAINTYCHECK.

Now, we aim to ensure that the VA and MA behave identically in the final loop. Notice that UNCERTAINTYCHECK-EGSS provides a weaker guarantee than UNCERTAINTYCHECK, when the returned result is CERTAIN. Specifically, when UNCERTAINTYCHECK-EGSS returns a result of CERTAIN, then Lemma 11 guarantees that $\|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \le d\tau$ for all $a\in\mathcal{A}$. While when the UNCERTAINTYCHECK returns a result of CERTAIN, then $\|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \le \tau$ for all $a\in\mathcal{A}$. Thus, we define a smaller good set $\mathcal{D}_d = \{\phi(s,a) : \|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \le d\tau\}$.

Redefine the VA's Q-function at iteration k as

$$\tilde{Q}_{k-1}(s,a) = \begin{cases} \tilde{w}_k^\top\phi(s,a) & \text{if } \phi(s,a)\in\mathcal{D}_d \\ Q_{\tilde{\pi}_{k-1}}(s,a) & \text{if } \phi(s,a)\notin\mathcal{D}_d \end{cases}$$

and VA's policy as

$$\tilde{\pi}_k(a|s) = \mathbb{1}\left(a = \arg\max_{\tilde{a}\in\mathcal{A}}\tilde{Q}_{k-1}(s,\tilde{a})\right).$$

Notice that in the final loop UNCERTAINTYCHECK-EGSS always returns a RESULT of CERTAIN, and thus we are sure that all $a\in\mathcal{A}$ for all the states encountered in the final loop are in the smaller good set $\mathcal{D}_d$. Thus, the VA's policy π_k would always be greedy w.r.t. $\tilde{w}_k^\top\phi$ in the final loop. This ensures that the VA and MA behave identically in the final loop.

Next we need show that we can bound $\|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)\|_\infty$ with this new definition of $\tilde{Q}_{k-1}$. First we state a slight modification of Lemma 8 that holds for the smaller good set $\mathcal{D}_d$

Lemma 12 (EGSS modified Lemma B.2 from Yin et al. (2021))**.** *Suppose that Assumption 1 holds. With all terms as defined earlier and $\theta > 0$. Then, with probability at least*

$$1 - 2C_{max}\exp(-2\theta^2(1-\gamma)^2 n)$$

for any $(s,a) \in (\mathcal{S} \times \mathcal{A})$ pair such that $\phi(s,a) \in \mathcal{D}_d$, we have

$$|\tilde{w}_k^\top \phi(s,a) - w_{\tilde{\pi}_{k-1}}^\top \phi(s,a)| \leq b\sqrt{\lambda d\tau} + \left(\epsilon + \frac{\gamma^{H+1}}{1-\gamma} + \theta\right)\sqrt{d\tau C_{max}} + \epsilon = \sqrt{d}\bar{\eta} := \eta_2$$

Proof. The proof is identical to that of Lemme B.2 from Yin et al. (2021) except τ is replaced with $d\tau$ everywhere, due to the weaker guarantee of UNCERTAINTYCHECK-EGSSas discussed above. □

Essentially we get an extra $\sqrt{d}$ factor due to the smaller good set $\mathcal{D}_d$. Since the VA's policy $\tilde{\pi}_k$ has access to the true Q-function $Q_{\tilde{\pi}_{k-1}}$ for all $\phi(s,a) \notin \mathcal{D}_d$, we can show that $\|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)\|_\infty$ can be bounded.

Proposition 13 (approximate value function bound for EGSS)**.** *Suppose that Assumption 1 holds. With all terms as defined earlier and $\theta > 0$. Then, with probability at least*

$$1 - 2C_{max}\exp(-2\theta^2(1-\gamma)^2 n)$$

we have

$$\|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)\|_\infty \leq \eta_2.$$

Proof. For any $(s,a) \in (\mathcal{S} \times \mathcal{A})$ such that $\phi(s,a) \in \mathcal{D}_d$, we have

$$|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)| \leq \eta_2 \tag{9}$$

by Proposition 13. While for any $(s,a) \in (\mathcal{S} \times \mathcal{A})$ such that $\phi(s,a) \notin \mathcal{D}_d$, we have

$$|\tilde{Q}_{k-1}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)| = |Q_{\tilde{\pi}_{k-1}}(s,a) - Q_{\tilde{\pi}_{k-1}}(s,a)| = 0 \tag{10}$$

□

Finally, it is left to show that $|\tilde{w}_k^\top \phi(\rho,a) - Q_{\tilde{\pi}_{k-1}}(\rho,a)|$ can be bounded for all $a \in \mathcal{A}$. Notice that lines 4-8 in CONFIDENT MC-LSPI run UNCERTAINTYCHECK-EGSS with state ρ as input until the returned status is CERTAIN. Recall that once UNCERTAINTYCHECK-EGSS returns a status of CERTAIN we know that $\rho \in \mathcal{D}_d$. Thus, we can immediately apply Lemma 12 to bound $\eta_2 \geq |\tilde{w}_k^\top \phi(\rho,a) - Q_{\tilde{\pi}_{k-1}}(\rho,a)|,\ \forall a \in \mathcal{A}$.

C.2 Default Action Vector (DAV) Method

In this section we prove some useful results for UNCERTAINTYCHECK-DAV. Fix a state $s \in \mathcal{S}$. As mentioned in the body, assume the action space can be decomposed as a product $\mathcal{A}^{(1:m)} = \mathcal{A}^{(1)} \times ... \times \mathcal{A}^{(m)}$ throughout this section. We call elements of $\mathcal{A}^{(1:m)}$ *action vectors*. First, UNCERTAINTYCHECK-DAV only iterates over $\sum_{i=1}^m |A^{(i)}|$ action vectors instead of all the action vectors like UNCERTAINTYCHECK does. Define the $\sum_{i=1}^m A^{(i)}$ sized set of modified default action vectors as $\bar{\mathcal{A}}^{(1:m)} = \{(a^{(i)}, \bar{a}^{(-i)}) : a^{(i)} \in \mathcal{A}^{(i)},\ i \in [m]\}$. Notice UNCERTAINTYCHECK-DAV iterates over all the actions in the set $a^{(1:m)} \in \bar{\mathcal{A}}^{(1:m)}$ and checks if any of them satisfy $\|\phi(s, a^{(1:m)})\|^2_{V_\mathcal{C}^{-1}} > \tau$. This of course achieves the goal of compute independent of $|\mathcal{A}^{(1:m)}|$, since there are only $\sum_{i=1}^m A^{(i)}$ action vectors in $\bar{\mathcal{A}}^{(1:m)}$ to iterate over now. Also, since only $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ that satisfy $\|\phi(s, a^{(1:m)})\|^2_{V_\mathcal{C}^{-1}} > \tau$ are added to the core set, we can still use Lemma 7 to bound the size of the core set by $C_{\max}$.

Now, we aim to ensure that the VA and MA behave identically in the final loop. Define the set of states for which all the modified default action vectors are in the good set as $\bar{\mathcal{S}} = \{s \in \mathcal{S} : \|\phi(s, a^{(1:m)})\|^2_{V_\mathcal{C}^{-1}} \leq \tau, \forall a^{(1:m)} \in \bar{\mathcal{A}}^{(1:m)}\}$. Redefine the VA's Q-function as

$$\tilde{Q}_{k-1}(s, a^{(1:m)}) = \begin{cases} \tilde{w}_k^\top \phi(s, a^{(1:m)}) & s \in \bar{\mathcal{S}} \\ Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)}). & s \in \mathcal{S} \backslash \bar{\mathcal{S}} \end{cases}$$

The VA's policy is

$$\tilde{\pi}_k(a^{(1:m)}|s) = \mathbb{1}\left(a^{(1:m)} = \underset{\tilde{a}^{(1:m)} \in \mathcal{A}^{(1:m)}}{\arg\max}\ \tilde{Q}_{k-1}(s, \tilde{a}^{(1:m)})\right).$$

Notice that in the final loop the check $\phi(s, (a^{(j)}, \bar{a}^{(-j)}))^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} \phi(s, (a^{(j)}, \bar{a}^{(-j)})) > \tau$ in UNCERTAINTYCHECK-DAV never returns TRUE, and thus we are sure that all $a^{(1:m)} \in \bar{\mathcal{A}}^{(1:m)}$ for all the states encountered in the final loop are in the good set. Notice that these states that satisfy this condition are state in $\bar{\mathcal{S}}$. Thus, the

VA's policy π_k would always be greedy w.r.t. $\tilde{w}_k^\top \phi$ in the final loop. This ensures that the VA and MA behave identically in the final loop.

Now we show that we can bound $\|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})\|_\infty$ with this new definition of $\tilde{Q}_{k-1}$. First we state a slight modification of Lemma 8 for $w_{\tilde{\pi}_{k-1}}^\top \phi$ instead of $Q_{\tilde{\pi}_{k-1}}$ which excludes the $\|w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})\|_\infty \leq \epsilon$ term in the proof of Lemma B.2 in (Yin et al., 2021).

Lemma 14 (Lemma B.2 in (Yin et al., 2021))**.** *Suppose that Assumption 4 holds. With all terms as defined earlier and $\theta > 0$. Then, with probability at least*

$$1 - 2C_{max} \exp(-2\theta^2 (1-\gamma)^2 n)$$

for any $(s, a^{(1:m)}) \in (\mathcal{S} \times \mathcal{A}^{(1:m)})$ pair such that $\phi(s, a^{(1:m)}) \in \mathcal{D}$, we have

$$|\tilde{w}_k(s, a^{(1:m)}) - w_{\tilde{\pi}_{k-1}}^\top(s, a^{(1:m)})| \leq b\sqrt{\lambda\tau} + \left(\epsilon + \frac{\gamma^{H-1}}{1-\gamma} + \theta\right)\sqrt{\tau C_{max}} := \bar{\eta}$$

The following Proposition gives us a bound on $\|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})\|_\infty$.

Proposition 15 (approximate value function bound for DAV)**.** *Suppose that Assumption 4 holds. With all terms as defined earlier and $\theta > 0$. Then, with probability at least*

$$1 - 2C_{max} \exp(-2\theta^2 (1-\gamma)^2 n)$$

we have

$$\|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})\|_\infty \leq \bar{\eta}(2m-1) + \epsilon := \eta_1.$$

Proof. For any $(s, a^{(1:m)}) \in (\bar{\mathcal{S}} \times \mathcal{A}^{(1:m)})$, we have

$$\begin{aligned}
&|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&= |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&= |\tilde{w}_k^\top \phi(s, a^{(1:m)}) \pm w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&\leq |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)})| + |w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&\leq |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)})| + \epsilon \\
&= |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)}) \pm (m-1)\tilde{w}_k^\top \phi(s, \bar{a}^{(1:m)}) \pm (m-1) w_{\tilde{\pi}_{k-1}}^\top \phi(s, \bar{a}^{(1:m)})| + \epsilon \\
&= \left| \left(\sum_{i=1}^{m} \tilde{w}_k^\top \phi(s, (a^{(i)}, \bar{a}^{(-i)})) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, (a^{(i)}, \bar{a}^{(-i)})) \right) + (m-1) \left[w_{\tilde{\pi}_{k-1}}^\top \phi(s, \bar{a}^{(1:m)})) - \tilde{w}_k^\top \phi(s, \bar{a}^{(1:m)}) \right] \right| + \epsilon \\
&\leq m\bar{\eta} + (m-1)\bar{\eta} + \epsilon \\
&= \bar{\eta}(2m-1) + \epsilon
\end{aligned} \tag{11}$$

where the second last inequality holds by Lemma 14 (because the features of all the state action pairs considered are in $\mathcal{D}$, since $s \in \bar{\mathcal{S}}$).

While for any $(s, a^{(1:m)}) \in ((\mathcal{S} \backslash \bar{\mathcal{S}}) \times \mathcal{A}^{(1:m)})$, we have

$$|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| = |Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| = 0 \tag{12}$$

□

Finally, it is left to show that $|\tilde{w}_k^\top \phi(\rho, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(\rho, a^{(1:m)})|$ can be bounded for all $a^{(1:m)} \in \mathcal{A}^{(1:m)}$. Notice that lines 4-8 in CONFIDENT MC-LSPI run UNCERTAINTYCHECK-DAV with state ρ as input until the returned status is CERTAIN. Recall that once UNCERTAINTYCHECK-DAV returns a status of CERTAIN we know that $\rho \in \bar{\mathcal{S}}$. Thus, we can immediately apply the result in Eq. (11) to bound $\eta_1 \geq |\tilde{w}_k^\top \phi(\rho, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(\rho, a^{(1:m)})|,\ \forall a^{(1:m)} \in \mathcal{A}^{(1:m)}$.

Algorithm 6 CONFIDENT MC-POLITEX

Input: initial state ρ, initial policy π_0, number of iterations K, threshold τ, number of rollouts n, length of rollout H
Globals: default action $\bar{a}$, regularization coefficient λ, discount γ, subroutine UNCERTAINTYCHECK
$\mathcal{C} \leftarrow \{(\rho, \bar{a}, \phi(\rho, \bar{a}), \text{NONE})\}$
status, result $\leftarrow$ UNCERTAINTYCHECK$(\rho, \mathcal{C}, \tau)$
while status $=$ UNCERTAIN **do**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\text{result}\}$
    status, result $\leftarrow$ UNCERTAINTYCHECK$(\rho, \mathcal{C}, \tau)$
end while
$z_q \leftarrow$ NONE, $\forall z \in \mathcal{C}$ ▷ Policy iteration starts $(*)$
for $k \in 1, \ldots, K$ **do**
    for $z \in \mathcal{C}$ **do**
        status, result $\leftarrow$ ROLLOUT$(n, H, \pi_{k-1}, z, \mathcal{C}, \tau)$
        if status $=$ DONE, **then** $z_q =$ result
        else $\mathcal{C} \leftarrow \mathcal{C} \cup \{\text{result}\}$ and **goto** line $(*)$
    end for
    $w_k \leftarrow (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \Phi_{\mathcal{C}}^\top q_{\mathcal{C}}$
    $\pi_k(a^{(1:m)}|s) \leftarrow\propto \prod_{i=1}^m \prod_{j=0}^{k-1} \exp\left(\alpha w_j^\top \phi_i(s, a^{(i)})\right)$.
end for
return $\bar{\pi}_{K-1} \sim \text{Unif}\{\pi_k\}_{k=0}^{K-1}$

C.3 Extending to Politex

Recall the above results where for the CONFIDENT MC-LSPI algorithm. The CONFIDENT MC-POLITEX algorithm can be found as Algorithm 6. It turns out the story for CONFIDENT MC-POLITEX is extremely similar and can be argued in nearly the same way. The main difference is that the policy used in CONFIDENT MC-POLITEX is different than in CONFIDENT MC-LSPI (line 17 in CONFIDENT MC-POLITEX is different from line 17 in CONFIDENT MC-LSPI). As such, we can no longer use Lemma 9 (since it relied on a greedy policy) and, thus cannot use Proposition 10 to bound the sub-optimality of the policy output by CONFIDENT MC-POLITEX. Next, we show there is a similar Lemma and Proposition that can derived for CONFIDENT MC-POLITEX.

Recall that we do not use clipping on the Q-functions in CONFIDENT MC-POLITEX, so that we can sample from the policy efficiently (Proposition 6). Importantly Proposition 6 only holds when Assumption 4 is satisfied. Thus, for the remainder of this section we will be working with the product action space $\mathcal{A}^{(1:m)}$. This means we must define the VA's Q-function differently from (Yin et al., 2021), by removing clipping from the case when $\phi(s, a^{(1:m)}) \in \mathcal{D}$.

$$\tilde{Q}_{k-1}(s, a^{(1:m)}) = \begin{cases} \tilde{w}_k^\top \phi(s, a^{(1:m)}) & \text{if } \phi(s, a^{(1:m)}) \in \mathcal{D} \\ Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)}) & \text{if } \phi(s, a^{(1:m)}) \notin \mathcal{D} \end{cases}$$

Then the VA's policy is

$$\tilde{\pi}_k(a^{(1:m)}|s) \propto \exp\left(\alpha \sum_{j=0}^{k-1} \tilde{Q}_j(s, a^{(1:m)})\right). \tag{13}$$

Also, due to no clipping, the sequence of Q-functions during policy iteration is now in the $[-\eta, (1-\gamma)^{-1} + \eta]$ interval, where $\eta \geq \|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})\|_\infty$. We now restate Lemma D.1 from Yin et al. (2021) which bounds the mixture policy output by Politex for an arbitrary sequence of Q-functions Since we do not use clipping the theorem is slightly modified (we replace the interval $[0, (1-\gamma)^{-1}]$ with a general interval $[a, b],\ a, b \in \mathbb{R}$, which can be extracted from the calculations in Szepesvári (2022b)).

Lemma 16 (modified Lemma D.1 in Yin et al. (2021) also in Szepesvári (2022b))**.** *Given an initial policy π_0, a sequence of functions $Q_k : \mathcal{S} \times \mathcal{A}^{(1:m)} \to [a, b],\ k \in [K-1], a, b \in \mathbb{R}$, and $Q_{\pi^*} \in [0, 1/(1-\gamma)]$, construct a sequence of policies $\pi_1, ..., \pi_{K-1}$ according to (Eq.* (13)*) with $\alpha = 1/(b-a)\sqrt{\frac{2\log(|\mathcal{A}^{(1:m)}|)}{K}}$, then, for any $s \in \mathcal{S}$, the mixture policy $\bar{\pi}_{K-1} \sim$ Unif$\{\pi_k\}_{k=0}^{K-1}$ satisfies*

$$V^*(s) - V_{\bar{\pi}_K}(s) \leq \frac{b-a}{(1-\gamma)}\sqrt{\frac{2\log(|\mathcal{A}^{(1:m)}|)}{K}} + \frac{2\max_{0\leq k\leq K-1}\|Q_k - Q_{\pi_k}\|_\infty}{1-\gamma} \tag{14}$$

Notice that the above result suggests we just need to control the term $\|Q_k - Q_{\pi_k}\|_\infty$. For the VA this is $\|\tilde{Q}_k - Q_{\tilde{\pi}_k}\|_\infty$ and as we have already seen, this can be bounded using the high probability bound on policy evaluation for UNCERTAINTY CHCEK WITH DAV (Proposition 15) and UNCERTAINTY CHCEK WITH EGSS (Proposition Proposition 13). Using Lemma 16 instead of Lemma D.1 in Yin et al. (2021), one can extract another slightly modified result from Yin et al. (2021).

Proposition 17 (equation (D.8) in Yin et al. (2021))**.** *With all terms as defined earlier. Define* $\eta \geq \|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})\|_\infty$. *Suppose* $\eta \geq |\tilde{w}_k^\top \phi(\rho, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(\rho, a^{(1:m)})|_\infty$, $\forall a^{(1:m)} \in \mathcal{A}^{(1:m)}$. *Then, if the VA and MA behave identically in the final loop, with probability at least* $1 - 4KC_{max}^2\exp(-2\theta^2(1-\gamma)^2 n)$ *we have*

$$V^*(s) - V_{\bar{\pi}_{K-1}}(\rho) \leq \frac{b-a}{(1-\gamma)}\sqrt{\frac{2\log(|\mathcal{A}^{(1:m)}|)}{K}} + \frac{4\eta}{1-\gamma} \tag{15}$$

Notice, that we require the same three things as in the CONFIDENT MC-LSPI case (Proposition 10). We need a bound on $\|\tilde{Q}_{k-1}(s, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})\|_\infty$. We need a bound on $|\tilde{w}_k^\top \phi(\rho, a^{(1:m)}) - Q_{\tilde{\pi}_{k-1}}(\rho, a^{(1:m)})|_\infty$, $\forall a^{(1:m)} \in \mathcal{A}^{(1:m)}$. We need to ensure that the VA and MA behave identically in the final loop. Then, we can get a bound on the sub-optimality of the MA's output policy $\bar{\pi}_{K-1}$. Using the same steps as in the previous sections, one can verify that indeed, CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-DAV or UNCERTAINTYCHECK-EGSS does satisfy the above three conditions, with $\eta = \eta_1$ (η_1 as defined in Proposition 15) and $\eta = \eta_2$ (η_2 as defined in Proposition 13) respectively.

We bound $|\mathcal{A}^{(1:m)}| = \prod_{i=1}^m |A^{(i)}| \leq \max_{i\in[m]} |\mathcal{A}^{(i)}|$. We can replace $b-a$ with $1/(1-\gamma) + 2\eta$, since $w^\top \phi(s, a^{(1:m)}) \in [-\eta, (1-\gamma)^{-1} + \eta]$, $\forall (s \times a^{(1:m)}) \in (\mathcal{S} \times \mathcal{A}^{(1:m)})$ in the final loop for the same event which holds with probability at least $1 - 4KC_{\max}^2\exp(-2\theta^2(1-\gamma)^2 n)$ in Proposition 17. Applying Proposition 17 we get with probability at least $1 - 4KC_{\max}^2\exp(-2\theta^2(1-\gamma)^2 n)$ that

$$V^*(s) - V_{\bar{\pi}_{K-1}}(\rho) \leq \left(\frac{1}{(1-\gamma)^2} + \frac{2\eta}{(1-\gamma)}\right)\sqrt{\frac{2m\log(\max_{i\in[m]}|\mathcal{A}^{(i)}|)}{K}} + \frac{4\eta}{1-\gamma}. \tag{16}$$

D KERNEL SETTING

Define $q_k(s, a^{(1:m)})$ as the estimated rollout value for $(s, a^{(1:m)}) \in \mathcal{C}$ in round $k \in [K]$ of policy iteration, and $q_k = [q_k(s, a^{(1:m)})]_{(s,a^{(1:m)})\in\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}$ as the vector containing all rollout results at round k, using some fixed ordering of $\mathcal{C}$. In round k of policy iteration we need to compute the ridge estimate $\hat{Q}_k$ using q_k as least squares targets,

$$\hat{Q}_k = \operatorname*{arg\,min}_{Q\in\mathcal{H}} \sum_{(s,a^{(1:m)})\in\mathcal{C}} (Q(s, a^{(1:m)}) - q_k(s, a^{(1:m)}))^2 + \lambda\|Q\|_{\mathcal{H}}^2 = (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda \mathbf{I}_{\mathcal{H}})^{-1}\Phi_{\mathcal{C}}^\top q_k \tag{17}$$

Here, $\mathbf{I}_{\mathcal{H}} : \mathcal{H} \to \mathcal{H}$ is the identity mapping, $\Phi_{\mathcal{C}}$ can be formally defined as a map $\Phi_{\mathcal{C}} : \mathcal{H} \to \mathbb{R}^{|\mathcal{C}|}, f \mapsto [f(s, a^{(1:m)})]_{(s,a^{(1:m)})\in\mathcal{C}}$, $f \in \mathcal{H}$; and $\Phi_{\mathcal{C}}^\top : \mathbb{R}^{|\mathcal{C}|} \to \mathcal{H}$ is the adjoint of $\Phi_{\mathcal{C}}$.

Using the 'kernel trick' we express the estimator as follows

$$\hat{Q}_k = \Phi_{\mathcal{C}}^\top (K_{\mathcal{C}} + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} q_k \tag{18}$$

where $K_{\mathcal{C}} = \Phi_{\mathcal{C}}\Phi_{\mathcal{C}}^\top \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ is the kernel matrix. Lastly, we can evaluate for any $(s, a^{(1:m)}) \in \mathcal{S} \times \mathcal{A}^{(1:m)}$:

$$\hat{Q}_k(s, a^{(1:m)}) = \mathbf{k}_{\mathcal{C}}(s, a^{(1:m)})^\top (K_{\mathcal{C}} + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} q_k \tag{19}$$

where we defined $\mathbf{k}_{\mathcal{C}}(s, a^{(1:m)}) = [\mathbf{k}(s, a^{(1:m)}, s', a'^{(1:m)})]_{(s',a'^{(1:m)})\in\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}$ (using the same fixed ordering of $\mathcal{C}$). Importantly, the last display only involves finite-dimensional quantities that can be computed from kernel evaluations. Moreover, since $\mathbf{k}(s, a^{(1:m)}, s', a'^{(1:m)}) = \sum_{j=1}^m \mathbf{k}_j(s, a^{(j)}, s', a'^{(j)})$ we can write

$$\hat{Q}_k(s, a^{(1:m)}) = \sum_{j=1}^m \hat{Q}_{k,j}(s, a^{(j)}) \tag{20}$$

$$\hat{Q}_{k,j}(s, a^{(j)}) := \mathbf{k}_{j,\mathcal{C}}(s, a^{(j)})^\top (K_{\mathcal{C}} + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} q_k \tag{21}$$

Algorithm 7 CONFIDENT KERNEL MC-LSPI/POLITEX

Input: initial state ρ, initial policy π_0, number of iterations K, threshold τ, number of rollouts n, length of rollout H
Globals: default action $\bar{a}$, regularization coefficient λ, discount γ, subroutine UNCERTAINTYCHECK, kernel $\mathbf{k}$
$\mathcal{C} \leftarrow \{(\rho, \bar{a}, \phi(\rho, \bar{a}), \text{NONE})\}$
status, result $\leftarrow$ UNCERTAINTYCHECK$(\rho, \mathcal{C}, \tau)$
while status $=$ UNCERTAIN **do**
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\text{result}\}$
  status, result $\leftarrow$ UNCERTAINTYCHECK$(\rho, \mathcal{C}, \tau)$
end while
$z_q \leftarrow$ NONE, $\forall z \in \mathcal{C}$ ▷ Policy iteration starts $(*)$
for $k \in 1, \ldots, K$ **do**
  for $z \in \mathcal{C}$ **do**
    status, result $\leftarrow$ ROLLOUT$(n, H, \pi_{k-1}, z, \mathcal{C}, \tau)$
    if status $=$ DONE, **then** $z_q =$ result
    else $\mathcal{C} \leftarrow \mathcal{C} \cup \{\text{result}\}$ and **goto** line $(*)$
  end for
  $\hat{Q}_k = \Phi_{\mathcal{C}}^\top (K_{\mathcal{C}} + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} q_k$
  $\pi_k(a^{(1:m)}|s) \leftarrow \begin{cases} \mathbb{1}\left(a^{(1:m)} = \underset{\tilde{a}^{(1:m)} \in \mathcal{A}^{(1:m)}}{\arg\max}\ \hat{Q}_k(s, \tilde{a}^{(1:m)})\right) & \text{LSPI} \\ \propto \prod_{i=1}^{m} \prod_{j=0}^{k-1} \exp\left(\alpha \hat{Q}_{j,i}(s, a^{(i)})\right). & \text{Politex} \end{cases}$
end for
return π_{K-1} for LSPI, or $\bar{\pi}_{K-1} \sim \text{Unif}\{\pi_k\}_{k=0}^{K-1}$ for Politex

where $\mathbf{k}_{j,\mathcal{C}}(s, a^{(j)}) = [\mathbf{k}_j(s, a^{(j)}, s', a'^{(j)})]_{(s', a'^{(1:m)}) \in \mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}$. Since the term $(K_{\mathcal{C}} + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} q_k$ is fixed for each j, we can still compute the maximizer independently for each $j \in [m]$ by iterating over all actions. This allows us to define CONFIDENT KERNEL MC-LSPI/POLITEX (Algorithm 7), which makes use of Eq. (21) in line 17 for calculating the policy.

The second quantity required by the algorithm is the squared norm $\|\phi(s, a^{(1:m)})\|^2_{(\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda \mathbf{I}_{\mathcal{H}})^{-1}}$, where now $\phi(s, a^{(1:m)}) = \mathbf{k}(s, a^{(1:m)}, \cdot, \cdot) \in \mathcal{H}$. A direct extension of the Woodbury formula to infinite vector spaces shows that

$$\lambda(\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda \mathbf{I}_{\mathcal{H}})^{-1} = \mathbf{I}_{\mathcal{H}} - \Phi_{\mathcal{C}}^\top (K_{\mathcal{C}} + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} \Phi_{\mathcal{C}} \tag{22}$$

Therefore the feature norm can be written using finite-dimensional quantities:

$$\|\phi(s, a^{(1:m)})\|^2_{(\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda \mathbf{I}_{\mathcal{H}})^{-1}} = \frac{1}{\lambda}\left(\mathbf{k}(s, a^{(1:m)}, s, a^{(1:m)}) - \mathbf{k}_{\mathcal{C}}(s, a^{(1:m)})^\top (K_{\mathcal{C}} + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} \mathbf{k}_{\mathcal{C}}(s, a^{(1:m)})\right) \tag{23}$$

With this, we can define UNCERTAINTYCHECK-K-DAV (Algorithm 8) which makes use of Eq. (23).

Algorithm 8 UNCERTAINTY CHECK WITH KERNEL-DEFAULT ACTION VECTOR (K-DAV)

Input: state s, core set $\Phi_{\mathcal{C}}$, threshold τ.
Globals: number of action components m.
for $j \in [m]$ **do**
  for $a^{(j)} \in \mathcal{A}^{(j)}$ **do**
    $\tilde{a} \leftarrow (a^{(j)}, \bar{a}^{(-j)})$
    if $\frac{1}{\lambda}\left(\mathbf{k}(s, \tilde{a}, s, \tilde{a}) - \mathbf{k}_{\mathcal{C}}(s, \tilde{a})^\top (\Phi_{\mathcal{C}} \Phi_{\mathcal{C}}^\top + \lambda \mathbf{I}_{|\mathcal{C}|})^{-1} \mathbf{k}_{\mathcal{C}}(s, \tilde{a})\right) > \tau$ **then**
      result $\leftarrow (s, \tilde{a}, \phi(s, \tilde{a}), \text{NONE})$
      return UNCERTAIN, result
    end if
  end for
end for
return CERTAIN, NONE

Analysis Our goal next is to extend the analysis used in the finite case to the kernel case, carefully arguing that the linear dimension d can be replaced by a more benign quantity. A common complexity measure is the total information gain, which we define as follows:

$$\Gamma(\lambda; \mathcal{C}) = \log\det(\Phi_{\mathcal{C}}^{\top}\Phi_{\mathcal{C}} + \lambda \mathbf{I}_{\mathcal{H}}) - \log\det(\lambda \mathbf{I}_{\mathcal{H}}) \tag{24}$$

Note that we can compute $\Gamma(\lambda; \mathcal{C})$ for any given core set $\mathcal{C}$. In the kernel case, we can compute $\Gamma(\lambda; \mathcal{C}) = \log\det(\mathbf{I}_{|\mathcal{C}|} + \lambda^{-1} K_{\mathcal{C}})$ using similar arguments as before.

The maximum information gain is

$$\Gamma_t(\lambda) = \max_{\mathcal{C}:|\mathcal{C}|=t} \Gamma(\lambda; \mathcal{C}).$$

It serves as a complexity measure in the bandit literature and can be bounded for many kernels of interests (Srinivas et al., 2009; Vakili et al., 2021). Following Du et al. (2021); Huang et al. (2021b), we further define the *critical information gain* for any fixed constant $c > 0$,

$$\tilde{\Gamma}(\lambda, c) = \max\{t \geq 1 : ct \leq \Gamma_t(\lambda)\}. \tag{25}$$

Note that the proof of (Yin et al., 2021, Lemma 5.1) implies that $\log(1+\tau)|C| \leq \Gamma_{|C|}(\lambda)$ Thus, $|\mathcal{C}| \leq C_{\max} = \tilde{\Gamma}(\lambda, \log(1+\tau))$

Since the dimension d enters our bounds only through $C_{\max}$ we can immediately get a query complexity bound for the kernelized algorithm in terms of $\tilde{\Gamma}$. For the finite-dimensional case, (Yin et al., 2021, Lemma 5.1) shows that $\tilde{\Gamma} \leq \mathcal{O}(d)$, recovering the previous bound.

E PROOFS OF THEOREMS

We make a remark on the query complexity of CONFIDENT MC-LSPI and CONFIDENT MC-POLITEX. From Lemma 7 we know the core set size is bounded by $C_{\max} = \tilde{\mathcal{O}}(d)$. The total number of times Policy iteration is restarted (restart means line 14 in CONFIDENT MC-LSPI or CONFIDENT MC-POLITEX is reached) is thus at most $C_{\max}$. Each run of policy iteration can take as much as K iterations. In each iteration ROLLOUT is run at most $C_{\max}$ times. ROLLOUT does n rollouts of length H which queries the simulator once for each step. In total the number of queries performed by CONFIDENT MC-LSPI or CONFIDENT MC-POLITEXis bounded by $C_{\max}^2 KnH$. This equation is used to calculate the query cost for the different variants of CONFIDENT MC-LSPI, once all the parameter values have been calculated. Since, the only difference between CONFIDENT KERNEL MC-LSPI/POLITEX and CONFIDENT MC-LSPI or CONFIDENT MC-POLITEX is how the policy is calculated (lines 16-17 in each of the algorithms), thus we can use the same expression as above ($C_{\max}^2 KnH$) to bound the query complexity of CONFIDENT KERNEL MC-LSPI/POLITEX , with $C_{\max} = \tilde{\Gamma}(\lambda, c)$.

E.1 Proof of Theorem 1

Plugging in $\eta = \eta_2$ (η_2 as defined in Proposition 13) into Proposition 10. Suppose Assumptions 1 to 3 are satisfied with $\epsilon = 0$. By choosing appropriate parameters according to δ and κ, we can ensure that with probability at least $1 - \delta$ that the policy output by CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-EGSS , π_{K-1} satisfies:

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \kappa,$$

with the following parameter settings

$$\begin{aligned}
\tau &= 1 \\
\lambda &= \frac{\kappa^2(1-\gamma)^4}{1024b^2d} \\
\theta &= \frac{\kappa(1-\gamma)^2}{32\sqrt{d}\sqrt{C_{\max}}} \\
H &= \frac{\log\left(32\sqrt{C_{\max}}\sqrt{d}\right) - \log\left(\kappa(1-\gamma)^3\right)}{1-\gamma} - 1 \\
K &= \frac{\log\left(\frac{1}{\kappa(1-\gamma)^2}\right) + \log(8)}{1-\gamma} + 1 \\
n &= \frac{\log(\delta) - \log(4KC_{\max}^2)}{2\theta^2(1-\gamma)^2} \\
C_{\max} &= \frac{e}{e-1}\frac{1+\tau}{\tau}d\left(\log(1+\frac{1}{\tau}) + \log(1+\frac{1}{\lambda})\right)
\end{aligned}$$

with computational cost of poly$(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ and query cost $\mathcal{O}\left(\frac{d^4}{\kappa^2(1-\gamma)^8}\right)$

If $\epsilon > 0$, then by choosing parameters as above, with $\kappa = \frac{32\epsilon d}{(1-\gamma)^2}(1+\log(b^2\epsilon^{-2}d^{-1}))^{1/2}$, we can ensure that with probability of at least $1-\delta$ that π_{K-1} satisfies:

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \frac{64\epsilon d}{(1-\gamma)^2}(1+\log(1+b^2\epsilon^{-2}d^{-1}))^{1/2}$$

with computational cost of poly$(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ and query cost $\mathcal{O}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$

E.2 Proof of Theorem 2

Plugging in $\eta = \eta_1$ (η_1 as defined in Proposition 15) into Proposition 10. Suppose Assumptions 2 and 4 are satisfied with $\epsilon = 0$. By choosing appropriate parameters according to δ and κ, we can ensure that with probability at least $1-\delta$ that the policy output by CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-EGSS , π_{K-1} satisfies:

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \kappa,$$

with the following parameter settings

$$\begin{aligned}
\tau &= 1 \\
\lambda &= \frac{\kappa^2(1-\gamma)^4}{1024b^2(2m-1)^2} \\
\theta &= \frac{\kappa(1-\gamma)^2}{32(2m-1)\sqrt{C_{\max}}} \\
H &= \frac{\log\left(32\sqrt{C_{\max}}(2m-1)\right) - \log\left(\kappa(1-\gamma)^3\right)}{1-\gamma} - 1 \\
K &= \frac{\log\left(\frac{1}{\kappa(1-\gamma)^2}\right) + \log(8)}{1-\gamma} + 1 \\
n &= \frac{\log(\delta) - \log(4KC_{\max}^2)}{2\theta^2(1-\gamma)^2} \\
C_{\max} &= \frac{e}{e-1}\frac{1+\tau}{\tau}d\left(\log(1+\frac{1}{\tau}) + \log(1+\frac{1}{\lambda})\right)
\end{aligned}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ and query cost $\mathcal{O}\left(\frac{m^2d^3}{\kappa^2(1-\gamma)^8}\right)$

If $\epsilon > 0$, then by choosing parameters as above, with $\kappa = \frac{32\epsilon\sqrt{d}m}{(1-\gamma)^2}(1+\log(b^2\epsilon^{-2}d^{-1}))^{1/2}$, we can ensure that with probability of at least $1-\delta$ that π_{K-1} satisfies:

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \frac{128\epsilon\sqrt{d}m}{(1-\gamma)^2}(1+\log(1+b^2\epsilon^{-2}d^{-1}))^{1/2}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ and query cost $\mathcal{O}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$

E.3 Proof of Theorem 4 + UncertaintyCheck-EGSS case

Plugging in $\eta = \eta_1$ when UncertaintyCheck-DAV is used (η_1 as defined in Proposition 15) and $\eta = \eta_2$ when UncertaintyCheck-EGSS is used (η_2 as defined in Proposition 13) into Eq. (16). Setting $\zeta = 2m-1$ when UncertaintyCheck-DAV is used, and $\zeta = \sqrt{d}$ when UncertaintyCheck-EGSS is used. Suppose Assumptions 2 and 4 are satisfied with $\epsilon = 0$. By choosing appropriate parameters according to δ and κ, we can ensure that with probability at least $1-\delta$ that the policy output by Confident MC-Politex $\bar{\pi}_{K-1}$ satisfies:

$$V^*(\rho) - V_{\bar{\pi}_{K-1}}(\rho) \leq \kappa,$$

with the following parameter settings

$$\begin{aligned}
\tau &= 1 \\
\lambda &= \frac{\kappa^2(1-\gamma)^2}{576b^2\zeta^2} \\
\theta &= \frac{\kappa(1-\gamma)}{24\zeta\sqrt{C_{\max}}} \\
H &= \frac{\log\left(24\sqrt{C_{\max}}\zeta\right) - \log\left(\kappa(1-\gamma)^2\right)}{1-\gamma} - 1 \\
K &= 2m\log(A)\left(\frac{4}{\kappa^2(1-\gamma)^4} + \frac{3}{\kappa(1-\gamma)^2} + \frac{9}{16}\right) \\
n &= \frac{\log(\delta) - \log(4KC_{\max}^2)}{2\theta^2(1-\gamma)^2} \\
C_{\max} &= \frac{e}{e-1}\frac{1+\tau}{\tau}d\left(\log(1+\frac{1}{\tau}) + \log(1+\frac{1}{\lambda})\right)
\end{aligned}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ and query cost $\mathcal{O}\left(\frac{m\zeta^2d^3}{\kappa^4(1-\gamma)^9}\right)$

If $\epsilon > 0$, then by choosing parameters as above, with $\kappa = \frac{16\epsilon\sqrt{d}\zeta}{(1-\gamma)}(1+\log(b^2\epsilon^{-2}d^{-1}))^{1/2}$, we can ensure that with probability of at least $1-\delta$ that $\bar{\pi}_{K-1}$ satisfies:

$$V^*(\rho) - V_{\bar{\pi}_{K-1}}(\rho) \leq \frac{32\epsilon\sqrt{d}\zeta}{1-\gamma}(1+\log(1+b^2\epsilon^{-2}d^{-1}))^{1/2}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ and query cost $\mathcal{O}\left(\frac{md}{\epsilon^4(1-\gamma)^5}\right)$

E.4 Proof of Theorem 3

Plugging in $\eta = \eta_1$ (η_1 as defined in Proposition 15) into Proposition 10. Suppose Assumptions 2, 5 and 6 are satisfied with $\epsilon = 0$. By choosing appropriate parameters according to δ and κ, we can ensure that with probability at least $1-\delta$ that the policy output by Confident Kernel MC-LSPI π_{K-1} satisfies:

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \kappa,$$

with the following parameter settings

$$\begin{aligned}
\tau &= 1 \\
\lambda &= \frac{\kappa^2(1-\gamma)^4}{1024b^2(2m-1)^2} \\
\theta &= \frac{\kappa(1-\gamma)^2}{32(2m-1)\sqrt{C_{\max}}} \\
H &= \frac{\log\left(32\sqrt{C_{\max}}(2m-1)\right) - \log\left(\kappa(1-\gamma)^3\right)}{1-\gamma} - 1 \\
K &= \frac{\log\left(\frac{1}{\kappa(1-\gamma)^2}\right) + \log(8)}{1-\gamma} + 1 \\
n &= \frac{\log(\delta) - \log(4KC_{\max}^2)}{2\theta^2(1-\gamma)^2} \\
C_{\max} &= \tilde{\Gamma}(\lambda, \log(2))
\end{aligned}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, \tilde{\Gamma}(\lambda, \log(2)), \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ and query cost $\mathcal{O}\left(\frac{m^2\tilde{\Gamma}(\lambda,\log(2))^3}{\kappa^2(1-\gamma)^8}\right)$

If $\epsilon > 0$, then by choosing parameters as above, with $\kappa = \frac{16\epsilon m\sqrt{\tilde{\Gamma}(\lambda,\log(2))}}{(1-\gamma)^2}$, we can ensure that with probability of at least $1-\delta$ that π_{K-1} satisfies:

$$V^*(\rho) - V_{\pi_{K-1}}(\rho) \leq \frac{32\epsilon m\sqrt{\tilde{\Gamma}(\lambda, \log(2))}}{(1-\gamma)^2}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, \tilde{\Gamma}(\lambda, \log(2)), \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ and query cost $\mathcal{O}\left(\frac{\tilde{\Gamma}(\lambda,\log(2))^2}{\epsilon^2(1-\gamma)^4}\right)$

Theroem for CONFIDENT KERNEL MC-POLITEX combined with UNCERTAINTYCHECK-DAV As mentioned in the body we state the theorem bounding the sub-optimality of the policy output by CONFIDENT KERNEL MC-POLITEX combined with UNCERTAINTYCHECK-K-DAV.

Theorem 18 (CONFIDENT KERNEL MC-POLITEX DAV Sub-Optimality)**.** *Suppose Assumption Assumptions 2, 5 and 6 hold. Define $\tilde{\Gamma} := \tilde{\Gamma}(\lambda, \log(2))$. If $\epsilon = 0$, for any $\kappa > 0$, with probability at least $1-\delta$, the policy $\bar{\pi}_{K-1}$, output by* CONFIDENT KERNEL MC-POLITEX *combined with* UNCERTAINTYCHECK-K-DAV *satisfies*

$$V^*(\rho) - V_{\bar{\pi}_{K-1}}(\rho) \leq \kappa.$$

Further, the query cost is $\mathcal{O}\left(\frac{m^3\tilde{\Gamma}^3}{\kappa^4(1-\gamma)^9}\right)$ and computation cost is poly$(\sum_{i=1}^m |A^{(i)}|, \tilde{\Gamma}, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ If $\epsilon > 0$, then with probability at least $1-\delta$, the policy $\bar{\pi}_{K-1}$, output satisfies

$$V^*(\rho) - V_{\bar{\pi}_{K-1}}(\rho) \leq \frac{16\epsilon m\sqrt{\tilde{\Gamma}}}{1-\gamma}$$

Further, the query cost is $\mathcal{O}\left(\frac{m\tilde{\Gamma}}{\epsilon^4(1-\gamma)^5}\right)$ and computation cost is poly$(\sum_{i=1}^m |A^{(i)}|, \tilde{\Gamma}, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ The parameter settings for both cases are defined below.

E.5 Proof of Theorem 18

Plugging in $\eta = \eta_1$ (η_1 as defined in Proposition 15) into Eq. (16). Suppose Assumptions 2, 5 and 6 are satisfied with $\epsilon = 0$. By choosing appropriate parameters according to δ and κ, we can ensure that with probability at least $1-\delta$ that the policy output by CONFIDENT KERNEL MC-POLITEX combined with UNCERTAINTYCHECK-K-DAV , $\bar{\pi}_{K-1}$ satisfies:

$$V^*(\rho) - V_{\bar{\pi}_{K-1}}(\rho) \leq \kappa,$$

with the following parameter settings

$$\begin{aligned}
\tau &= 1 \\
\lambda &= \frac{\kappa^2(1-\gamma)^2}{576b^2(2m-1)^2} \\
\theta &= \frac{\kappa(1-\gamma)}{24(2m-1)\sqrt{C_{\max}}} \\
H &= \frac{\log\left(24\sqrt{C_{\max}}(2m-1)\right) - \log\left(\kappa(1-\gamma)^2\right)}{1-\gamma} - 1 \\
K &= 2m\log(A)\left(\frac{4}{\kappa^2(1-\gamma)^4} + \frac{3}{\kappa(1-\gamma)^2} + \frac{9}{16}\right) \\
n &= \frac{\log(\delta) - \log(4KC_{\max}^2)}{2\theta^2(1-\gamma)^2} \\
C_{\max} &= \tilde{\Gamma}(\lambda, \log(2))
\end{aligned}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, \tilde{\Gamma}(\lambda, \log(2)), \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ and query cost $\mathcal{O}\left(\frac{m^3\tilde{\Gamma}(\lambda,\log(2))^3}{\kappa^4(1-\gamma)^9}\right)$

If $\epsilon > 0$, then by choosing parameters as above, with $\kappa = \frac{8\epsilon m\sqrt{\tilde{\Gamma}(\lambda,\log(2))}}{(1-\gamma)}$, we can ensure that with probability of at least $1-\delta$ that $\bar{\pi}_{K-1}$ satisfies:

$$V^*(\rho) - V_{\bar{\pi}_{K-1}}(\rho) \leq \frac{16\epsilon m\sqrt{\tilde{\Gamma}(\lambda, \log(2))}}{1-\gamma}$$

with computational cost of poly$(\sum_{i=1}^m |A^{(i)}|, \tilde{\Gamma}(\lambda, \log(2)), \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ and query cost $\mathcal{O}\left(\frac{m\tilde{\Gamma}(\lambda,\log(2))}{\epsilon^4(1-\gamma)^5}\right)$

F EXAMPLES AND EXPERIMENTS

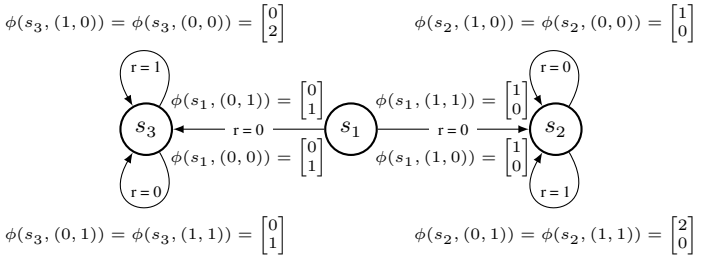

Figure 1: Illustration of Example 2.

F.1 Additive MDP, Cooperation Example

Example 2 (Coordination)**.** *Consider the MDP in Fig. 1, which can be verified to satisfy Assumption 4 with $\gamma = 1/2$ (proof in the next subsection). At every time step, two agents in the MDP take actions from $\mathcal{A}^{(1)} = \mathcal{A}^{(2)} = \{0,1\}$, and move to a next state together. The starting state is s_1 and by taking a joint action they move to s_2 or s_3, which are absorbing states and the agents will remain in them once they get there. It is easy to see that if we fix the policy for one of the agents in all states, the other agent will face a reduced MDP where the transitions only depends on the its actions. We will show that for two different policies followed by the second agent, the problem (the MDP) the first agent faces changes. More specifically, the best action for the first agent in s_1 is different in the resulting MDPs, which suggests that the first agent should coordinate with the second agent to achieve a higher value. It also shows that this example cannot be reduced to a product MDP, since in product MDPs the best action for each agent is irrespective of the behavior of the other agents.*

Assume two different policies $\pi_0, \pi_1 : \mathcal{S} \to \Delta_{\mathcal{A}^{(2)}}$ for the second agent, such that $\pi_0(s_i) = \delta_0, \pi_1(s_i) = \delta_1$ for $i \in [3]$ where δ_j for $j \in [2]$ is the Dirac delta distribution. Policy π_0 causes the joint policy π to get reward 1 in s_3 and get reward

0 in s_2, regardless of the policy followed by the first agent. The effect of following π_1 is exactly the opposite, meaning getting reward 1 in s_2 and 0 in s_3. Consequently, the optimal action for agent 1 depends on choosing π_0 or π_1 by the second agent. Therefore, agent 1 needs to coordinate its action with the second agent's policy to get the higher reward. This property, coordination with other agent's policy, cannot be modeled with separate MDPs since in those cases the optimal action for each agent only depends on the agent's MDP, and does not depend on the behavior of other agents. This example shows that the Assumption 4 is not limited to solving multiple MDPs with joint reward observation, and can model some cases where cooperation is needed.

Realizability

In this section we prove that the MDP in Fig. 1 satisfies Assumption 4. We start by showing that all the deterministic policies are realizable using the shown feature vectors. We use the weight vector $w_{(a_1^1,a_1^2),(a_2^1,a_2^2),(a_3^1,a_3^2)}$ for a deterministic policy that takes action vector (a_i^1, a_i^2) in state s_i for $i \in \{1,2,3\}$ and $a_i^1, a_i^2 \in \{0,1\}$. We also use $\cdot$ to show that the choice of an action in the respective state does not change the weight vector. One can verify that the following vectors satisfy realizability assumption:

$$w_{(\cdot,\cdot),(\cdot,0),(\cdot,0)} = \begin{bmatrix}0\\1\end{bmatrix}, \qquad w_{(\cdot,\cdot),(\cdot,0),(\cdot,1)} = \begin{bmatrix}0\\0\end{bmatrix},$$
$$w_{(\cdot,\cdot),(\cdot,1),(\cdot,0)} = \begin{bmatrix}1\\1\end{bmatrix}, \qquad w_{(\cdot,\cdot),(\cdot,1),(\cdot,1)} = \begin{bmatrix}1\\0\end{bmatrix}.$$

It remains to show that the non-deterministic policies are also realizable. For a policy π that takes action $(\cdot, 1)$ at s_2 with probability p_2, and action $(\cdot, 0)$ at s_3 with probability p_3, the realizable weight vector is:

$$w_\pi = \begin{bmatrix}p_2\\p_3\end{bmatrix}.$$

This holds since the choice of the action in s_1 does not change the weight vector in this example.

F.2 Experimental Results

We evaluate the performance of the proposed algorithms in a small grid world example as shown in Fig. 2. Each of four agents is placed in a 3x3 grid world. The agents obtain a +1 reward for reaching the goal state and a -1 reward in a 'trap' state. Reaching either the trap state or the reward state terminates the episode. Each agent has four actions to move to a neighboring cell. The selected action is applied with probability 0.95 while with 0.05 probability an action is chosen uniformly at random. The global reward is the sum of the agents rewards. Note that the individual rewards are not observed, therefore the example is different from four separate grid worlds.

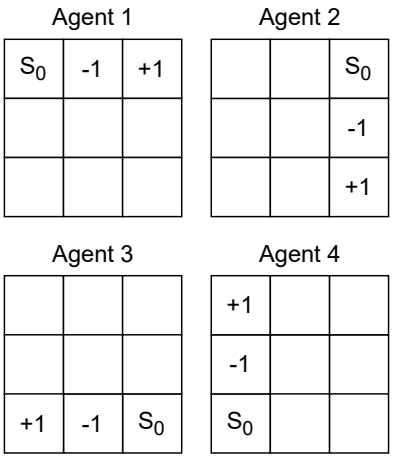

Figure 2: Four agent grid world.

We run each variant of the algorithm for 50 iterations ($K = 50$) without resets (the resets are mainly for simplicity of analysis). The discount factor is set to $\gamma = 0.8$, the regularization parameter is set to $\lambda = 10^{-5}$, for Politex we set $\alpha = 1$ and the rollout length is $H = 15$. The agents' individual features are one-hot encodings of agent, agent positions and actions which results in a feature of dimension $d = 4 \cdot 9 \cdot 4 = 144$. Note, however, that the joint MDP is *not* tabular, as the joint features, i.e. the sum over the agent features, are not one-hot vectors. In fact, the features are crucial for generalization as there are a total $9^4 = 9561$ joint states for all four agents combined.

Figure 3 shows two experiments with $n = 10$ and $n = 50$ rollouts. The plots show the performance of the policy estimate after each iteration averaged over 25 random seeds. We run both CONFIDENT MC-LSPI and CONFIDENT MC-POLITEX with EGSS (Algorithm 3) and DAV (Algorithm 4) uncertainty checks. In addition we compare to the NAIVE uncertainty check (Algorithm 5) that iterates over all $|A| = 4^4$ actions (Yin et al., 2021). Note that with 50 rollouts, both the EGSS and DAV variants perform essentially the same as NAIVE, despite the relaxed uncertainty bound. LSPI finds a good policy within at most five iterations. With only 10 rollouts, the final policy of CONFIDENT MC-LSPI converges to a suboptimal value on average. This can be understood as the data between iterations is not shared, and the noise from the Monte-Carlo estimates sometimes leads to a deteriorating in the policy

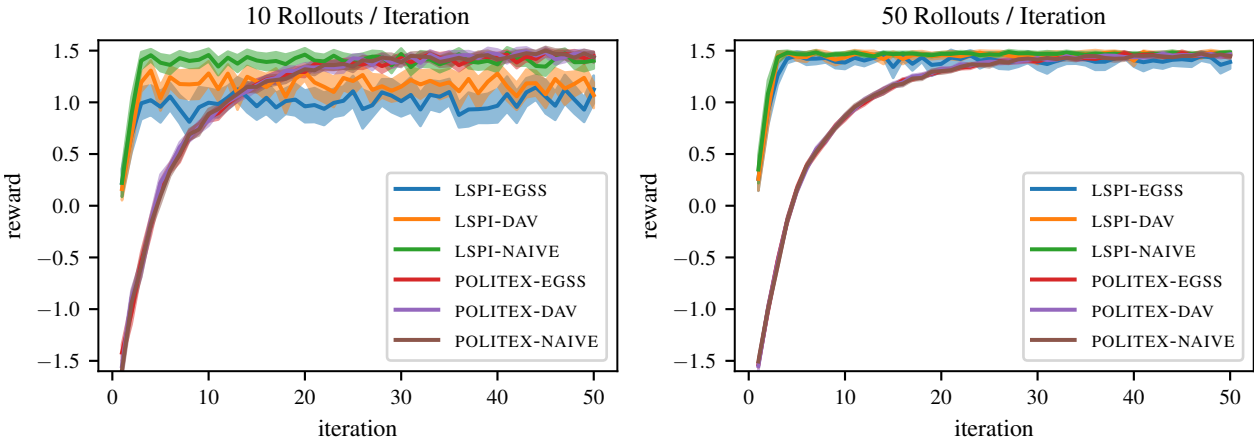

Figure 3: Numerical results on a grid world with four agents.

improvement step. With 50 rollouts per iteration, LSPI reliably finds the optimal policy in all MDPs. On the other hand, CONFIDENT MC-POLITEX is much more stable even with just 10 rollouts, but also requires more iterations to converge. This is expected because in CONFIDENT MC-POLITEX, the policy estimates from all iterations are averaged.