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

by

Volodymyr Tkachuk

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

A practical challenge in reinforcement learning is large 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 blow-up in the action space as the number of agents increases. 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. 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. For the special case where the feature decomposition is additive, we further improve the bounds if the dimension of the feature space is large relative to the number of additive terms in the feature decomposition. To the best of our knowledge, this work provides the first computationally efficient algorithms with theoretical guarantees for the reinforcement learning problem, when the action space is large and we have access to a simulator.

Preface

Chapters 4 to 8 of this thesis has been published as Volodymyr Tkachuk, Seyed Alireza Bakhtiari, Johannes Kirschner, Matej Jusup, Ilija Bogunovic, and Csaba Szepesvári. Efficient planning in combinatorial action spaces with applications to cooperative multi-agent reinforcement learning. In International Conference on Artificial Intelligence and Statistics, pages 6342–6370. PMLR, 2023.

The results in Sections 5.1 to 5.3 were obtained in collaboration with Seyed Alireza Bakhtiari. Discussions with Csaba Szepesvári were helpful in the developement of Chapter 7. Section 8 was a joint effort by Matej Jusup, Johannes Kirschner, Seyed Alireza Bakhtiari, Ilija Bogunovic, and myself. All remaining parts of this thesis are my original work.

Acknowledgements

I would like to thank my wonderful supervisor Csaba Szepesvári for his support throughout my research and his contagious energy towards life. I would also like to acknowledge Johannes Kirschner for his incredible patience and the never-ending assistance he has provided me throughout my two years of studies. I am grateful for my labmates Seyed Alireza Bakhtiari, Kushagra Chandak, and Chang Liu who I have had endless conversations about various research topics. Finally, I would like to thank the supportive group of students and faculty we have at RLAI, that have always been available whenever I needed any help.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Reinforcement learning (RL) is concerned with training agents to make near-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 near-optimal policy that maximizes the total reward. Efficiently computing optimal policies 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. When the agent has access to a simulator of the environment and its objective is to compute a near-optimal policy it is known as *planning*.

In this thesis, 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. 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 our realizability assumptions, previous approaches are not computationally efficient in the case where the action space is large, since their computational complexity scales with the number of actions (Yin et al., 2021; Hao et al., 2022; Weisz et al., 2022). 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 action space).

One prominent special case of this setting is multi-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 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. The additive structure also leads to an efficient implementation of the CONFIDENT MC-POLITEX algorithm that admits improved bounds in the misspecified setting.

A note on more recent work This thesis is based on extending the results from Yin et al. (2021) to large action spaces. At the time when the results in this thesis were developed, the work by Yin et al. (2021) provided the best results for the local access simulator setting. More recently, Weisz et al. (2022) have introduced a new algorithm CAPI-QPI-PLAN with better sample complexity and suboptimality guarantees than that of Yin et al. (2021). As such, it may be possible that our approach can be applied to the work of Weisz et al. (2022), extending it to large action spaces as well. However, due to the recency of CAPI-QPI-PLAN, we have not had the time to investigate this idea yet.

Chapter 2

Related Works

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) and 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), Lattimore et al. (2020), and Weisz et al. (2022). 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 encountered 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 large 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 large 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 guarantees 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 5.4 as a special case.

Multi-agent reinforcement learning (Busoniu et al., 2008; Zhang et al., 2021) can be understood as an RL problem with a large action space, due to the combinatorial joint action space of the agents, which has a large body of works on its own. Query complexity bounds focus mostly on the competitive setting, where each agent aims to maximize their own reward and the objective is to reach various forms of equilibrium (Nash, correlated, coarse-correlated). An example is in tabular Markov games (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. (2021); 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 reinforcement 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; however, they do not provide theoretical guarantees. 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.

Chapter 3

Problem Setting

We will assume the reader is familiar with the basics of Markov decision processes (MDPs) and recommend the excellent book by Puterman (2014) for any missing details. We consider reinforcement learning in an infinite-horizon Markov decision process (MDP) specified by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \gamma, \rho)$. Here, $\mathcal{S}$, $\mathcal{A}$ are the state and the action spaces, respectively, $\mathbb{P}$ is a transition kernel and r is a reward function. For convenience, we assume that the state and action spaces are finite. The transition kernel $\mathbb{P}$ maps state-action pairs to distributions over the state space: $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$, where $\Delta_{\mathcal{S}}$ denotes the set of probability measures over $\mathcal{S}$. Given a state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, the system transits to a new state $s' \sim \mathbb{P}(s, a)$. The reward function is $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$, $\gamma \in [0, 1)$ is the discount factor, and $\rho \in \mathcal{S}$ is the (deterministic) initial state.

A stationary memoryless policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ maps states to a distribution over $\mathcal{A}$. Starting from a state $s \in \mathcal{S}$ the policy interacts with the MDP $\mathcal{M}$ sequentially for time steps $t \in \mathbb{N}$. The interaction begins in $S_0 = s$, then an action is sampled $A_t \sim \pi(S_t)$ and a next state $S_{t+1} \sim \mathbb{P}(S_t, A_t)$. This interaction generates a trajectory $(S_i, A_i)_{i \in \mathbb{N}}$ with corresponding probability distribution over it, which we define as $\mathcal{P}_{\pi,s}$. We also define $\mathcal{P}_{\pi,s,a}$ to be the distribution over the trajectory when $A_0 = a$ deterministically.

The state-value function $v_\pi : \mathcal{S} \to \mathbb{R}$ of a policy π from a state $s \in \mathcal{S}$ is

$$v_\pi(s) = \mathbb{E}_{\pi,s}\left[\sum_{t\in\mathbb{N}} \gamma^t r(S_t, A_t)\right],$$

where the expectation $\mathbb{E}_{\pi,s}$ corresponds to the probability distribution $\mathcal{P}_{\pi,s}$. A policy π^* is *optimal* if $v_{\pi^*} = \max_\pi v_\pi$.

The action-value function (or 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

$$q_\pi(s, a) = \mathbb{E}_{\pi,s,a}\left[\sum_{t\in\mathbb{N}} \gamma^t r(S_t, A_t)\right].$$

where the expectation $\mathbb{E}_{\pi,s,a}$ corresponds to the probability distribution $\mathcal{P}_{\pi,s,a}$.

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)**.** *We assume there exists $b > 0, \epsilon \geq 0$, known to the algorithm designer, such that for each policy π there exists a weight vector $w_\pi \in \mathbb{R}^d, \|w_\pi\|_2 \leq b$ satisfying $\max_{s,a} |q_\pi(s, a) - w_\pi^\top \phi(s, a)| \leq \epsilon$.*

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 goal is to find a computational and query efficient algorithm that given a starting state $\rho \in \mathcal{S}$, and accuracy requirement $\kappa > 0$ returns a κ-optimal policy $\hat{\pi}$, i.e. $v_{\pi^*}(\rho) - v_{\hat{\pi}}(\rho) \leq \kappa$. 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 $r(s, a)$. With a local access simulator the planner begins by only having access to the initial state $\rho \in \mathcal{S}$, and encounters new states by querying the simulator. Crucially, the local access simulator restricts the planner's queries to only those states that have been previously encountered. A local access simulator can be implemented in practice by using checkpoints[1] when a new state is encountered, allowing the simulator to reload back to any of these checkpoints at a future time. Thus, the local access simulator is a more practical simulator when compared to the random access simulator (where the planner can query the simulator with any $s \in \mathcal{S}$ from the beginning) required by some previous works in planning (Lattimore et al., 2020).

In this thesis we are interested in problems with large action sets. 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 $[m] := \{1, \ldots, m\}$. We denote the joint action set by $\mathcal{A} = \mathcal{A}^{(1:m)} := \mathcal{A}^{(1)} \times ... \times \mathcal{A}^{(m)}$. 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 (if $|\mathcal{A}^{(i)}| > 1$ for all $i \in [m]$), which makes approaches designed for the single agent setting computationally intractable when the number of agents is large. We will revisit this example in Section 5.4 where we discuss how an additive feature decomposition leads to algorithms that scale polynomially in the number of agents m. We remark that

[1] a checkpoint is all relevant simulator information that is needed by the simulator to be reloaded to the state when the checkpoint was created.

prior work on cooperative multi-agent RL has focused on problem settings 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); however, they do not provide theoretical guarantees or do not consider the planning setting.

Chapter 4

Confident Monte-Carlo Least-Squares Policy Iteration

This chapter is devoted entirely to discussing the CONFIDENT MC-LSPI algorithm presented by Yin et al. (2021). Thus, this chapter does not contain any of our novel work. Since this thesis extends the work of CONFIDENT MC-LSPI to large action spaces, we use this chapter to explain the algorithm and its analysis to make future chapters more interpretable. We now describe how this chapter is structured. In Section 4.1 we introduce the CONFIDENT MC-LSPI algorithm and describe how it works. In Section 4.2 we present a result (Theorem 1) that provides guarantees on the performance of CONFIDENT MC-LSPI and its query and computation complexity. We highlight that the computational complexity scales linearly with $|\mathcal{A}|$, which is unacceptable for an efficient algorithm if the number of actions $|\mathcal{A}|$ is large. Lastly, in Section 4.2 we go through the analysis of CONFIDENT MC-LSPI and show the proof for Theorem 1. Importantly, many steps of the analysis will be reused in later chapters to argue about the performance of the novel algorithms we present, which will have computation that does not depend on $|\mathcal{A}|$ or depends only on poly$(\log |\mathcal{A}|)$.

4.1 Confident MC-LSPI

The Confident Monte-Carlo Least-Squares Policy Iteration (Confident MC-LSPI) algorithm proposed by Yin et al. (2021) was the first algorithm to provably output a near-optimal policy with polynomial query and computation complexity in all relevant problem parameters, while only using a local access simulator. Confident MC-LSPI is presented as Algorithm 1. We remark that Confident MC-LSPI as we have presented it here is structured to be more modular than the original version (LSPI case for algorithm 2 of Yin et al. (2021)) to ease the exposition of our contributions in future chapters. It should be noted that our Algorithm 1 is logically equivalent to Confident MC-LSPI of Yin et al. (2021). More precisely, Confident MC-LSPI with UncertaintyCheck-Naive (Algorithm 3) used for the UncertaintyCheck global subroutine is equivalent to the Confident MC-LSPI algorithm of Yin et al. (2021). Now we proceed to describe how Confident MC-LSPI combined with UncertaintyCheck-Naive works.

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 action-value. During each rollout, the UncertaintyCheck subroutine (Algorithm 3) determines if the core set should be expended because a feature direction is still underrepresented in it. 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*

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, $\forall z \in \mathcal{C}$ ▷ Policy iteration starts $(*)$
10: **for** $k = 1, \ldots, K$ **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}

set $\mathcal{C}$ with elements indexed by 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 by $q_{\mathcal{C}} = (z_q)_{z \in \mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}$. 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). Here, $\Phi_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}| \times d}$ is a matrix of all the features, as row vectors, from the tuples in the core set $\mathcal{C}$ stacked vertically. An improved policy π_k based on w_k is then obtained by being greedy with respect to $w^\top \phi(s, a)$ (line 17). Importantly, π_k is not explicitly computed in line 17 (which would depend on the number of states and actions), it is only computed at the states that an action needs to be taken (i.e. line 10 of ROLLOUT). The core set is initialized in lines 3-8 by adding the initial state ρ with a *default action* $\bar{a}$, so that there is at least one element in the core set to

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: $s_{i,0} \leftarrow z_s, a_{i,0} \leftarrow z_a$
4: Query the simulator, obtain $r_{i,0} \leftarrow r(s_{i,0}, a_{i,0})$, and the next state $s_{i,1}$
5: **for** $t = 1, ..., H$ **do**
6: status, result$\leftarrow$UNCERTAINTYCHECK$(s_{i,t}, \mathcal{C}, \tau)$
7: **if** status = UNCERTAIN **then**
8: **return** status, result
9: **end if**
10: Sample $a_{i,t} \sim \pi(\cdot|s_{i,t})$
11: 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 $s_{i,t+1} \sim \mathbb{P}(s_{i,t}, a_{i,t})$
12: **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

Algorithm 3 UNCERTAINTYCHECK-NAIVE

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

rollout from (line 3)[1].

Then, we continuously run the UNCERTAINTYCHECK algorithm until it stops returning a status of UNCERTAIN. Each time UNCERTAINTYCHECK returns UNCERTAIN we add the uncertain tuple (result variable) to the core set. This is to ensure that the final policy[2] 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)

[1]Although the initial state ρ can be added with any action for this chapter (as in done in Yin et al. (2021)), we specify a specific action $\bar{a}$, since this will be needed for our UNCERTAINTYCHECK-DAV algorithm in Section 5.4.

[2]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}.

are added to the core set (details in Section 4.3.3).

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 to get an estimate of the action-values at the state-action pair in z under policy π_{k-1}. The action-value 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 makes use of UNCERTAINTY-CHECK to determine if a rollout is *successful* or *unsuccessful*. If UNCERTAINTY-CHECK ever returns UNCERTAIN (line 6 of ROLLOUT) then ROLLOUT is *unsuccessful*, and it returns a status of UNCERTAIN and a corresponding uncertain tuple (result variable, line 8 in ROLLOUT). The uncertain tuple is added to the core set and policy iteration is restarted (line 14 in CONFIDENT MC-LSPI) and the value estimates for all the core set elements are reset to NONE (line 9 in CONFIDENT MC-LSPI). Important is that adding tuples to the core set in this way ensures that the size of the core set is bounded by $\mathcal{O}(d)$ (Lemma 4.1), and thus so are the number of restarts.

If UNCERTAINTYCHECK always returns CERTAIN (line 6 in ROLLOUT) then ROLLOUT is successful and it returns a status of DONE and an estimate of $q_{\pi_{k-1}}(z_s, z_a)$, which is assigned to z_q (line 13 in CONFIDENT MC-LSPI). 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 (Lemma 4.2).

Roughly speaking, the UNCERTAINTYCHECK should flag tuples $z = (z_s, z_a, z_\phi, z_q)$ as uncertain if their corresponding feature z_ϕ is sufficiently different from all the features in the core set $\{z_\phi : z \in \mathcal{C}\}$. As we will discuss in Section 5.3.3, a feature $\phi(s, a)$ with $s \in \mathcal{S}, a \in \mathcal{A}$ that satisfies

$$\phi(s,a)^\top (\Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I)^{-1} \phi(s,a) > \tau \tag{4.1}$$

is considered an uncertain feature. Here, recall that $\Phi_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}| \times d}$ is a matrix of all the features from the tuples in the core set stacked vertically.

The crucial assumption that makes the algorithm work is that regardless the policy, the same set of features can approximate well the action-value of the policy. This is why a finite policy-independent core set is sufficient at the end to approximate the action-value functions of all the policies that will be encountered in the algorithm. That the core set can also be kept small is a property of linear function approximation, in particular, a result due to Kiefer-Wolfowitz ensures this (Kiefer and Wolfowitz, 1960).

4.2 Theoretical Guarantees

The result that characterizes the performance of Confident MC-LSPI combined with UncertaintyCheck-Naive is summarized in the next theorem. In the theorem, we present slightly different results for the cases when there is no misspecification ($\epsilon = 0$) from the case when there is misspecification ($\epsilon > 0$). This is because in the case of misspecification, due to the misspecification, there is an unavoidable loss in terms of the optimality of the policy that can be obtained at polynomial cost (recall that the cost cannot depend on the number of states, which can be arbitrarily large). In fact, it is known that the unavoidable loss must scale with $\sqrt{d}\epsilon/(1-\gamma)$ when the computation is kept polynomial Weisz et al. (2022). Due to this unavoidable cost, in the presence of misspecification, we seek to set the parameters of the algorithm (i.e. τ, λ, H, K, n) such that the output policy's error is only slightly larger than the unavoidable loss. Specifically, in the Theorem below, since the misspecification is $\frac{32\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}$ we have set the parameters τ, λ, H, K, n such that all remaining terms in the bound of $v^*(\rho) - v_{\pi_{K-1}}(\rho)$ are also bounded by $\frac{32\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}$. Thus giving the final bound of

$$v^*(\rho) - v_{\pi_{K-1}}(\rho) \leq 2 \cdot \frac{32\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

Theorem 1 (CONFIDENT MC-LSPI Suboptimality (Theorem 5.1 in (Yin et al., 2021))). *Suppose Assumptions 1 and 2 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-LSPI *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.1) such that with probability at least $1-\delta$ the policy π_{K-1} returned by* CONFIDENT MC-LSPI *combined with* UNCERTAINTYCHECK-NAIVE *satisfies*

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

The query complexity is $\tilde{\mathcal{O}}\left(\frac{d^3}{\kappa^2(1-\gamma)^8}\right)$ and computation complexity is

$$poly\left(|\mathcal{A}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right).$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-LSPI *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.1) such that with probability at least $1 - \delta$ the policy π_{K-1} satisfies*

$$v^*(\rho) - v_{\pi_{K-1}}(\rho) \leq \tfrac{64\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

The query complexity is $\tilde{\mathcal{O}}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$ and the computation complexity is

$$poly\left(|\mathcal{A}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\epsilon}, \log\left(\tfrac{1}{\delta}\right), \log(1+b)\right).$$

Important to notice is that the computation complexity has a poly($|\mathcal{A}|$) dependence. This dependence is unacceptable for an efficient algorithm when the number of actions $|\mathcal{A}|$ is large. As such, in the remaining chapters we will be focused on how to remove this dependence, while still providing reasonable suboptimality guarantees of the output policy. We present the proof of Theorem 1 in the next section.

4.3 Analysis

In this section we prove Theorem 1. This involves arguing that Algorithm 1 is query and computationally efficient, while also providing suitable guarantees on the opti-

mality of its output policy. The proof is borrowed directly from Yin et al. (2021). We state many of the intermediate results without proof (and direct the reader to proofs in Yin et al. (2021)) as the intention of this chapter is to give a general idea for the steps involved. Thus, we emphasize the conceptual understanding of the proof, which will be useful for later chapters. The proof will proceed in these steps:

1. In Section 4.3.1 we show a bound on the number of queries needed by Confident MC-LSPI combined with UncertaintyCheck-Naive.
2. In Section 4.3.2 we show a bound on the computational complexity of Confident MC-LSPI combined with UncertaintyCheck-Naive.
3. In Section 4.3.3 we show the policy π_{K-1} output by Confident MC-LSPI combined with UncertaintyCheck-Naive is nearly optimal.
4. In Section 4.3.4 we combine the above three parts to prove Theorem 1.

4.3.1 Query Complexity Bound

We first state a bound on the size of the core set, which will be helpful in bounding the query complexity later.

Lemma 4.1 (Bound on Core Set Size (Lemma 5.1 in (Yin et al., 2021))). *Suppose Assumption 2 holds, and let $\tau, \lambda > 0$. Assume that a set $\mathcal{C}$ is constructed iteratively by processing tuples z of the form $z = (z_s, z_a, z_\phi, z_q)$ with feature $z_\phi = \phi(z_s, z_a) \in \mathbb{R}^d$ iteratively and adding a tuple z to $\mathcal{C}$ only if $z_\phi^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} z_\phi > \tau$ where $\Phi_\mathcal{C}$ is the matrix constructed from the feature components of the tuples in $\mathcal{C}$ at the time of processing z. Then, the size of set $\mathcal{C}$ will never get larger than*

$$C_{\max} := \frac{e}{e-1} \frac{1+\tau}{\tau} d \left(\log(1 + \frac{1}{\tau}) + \log(1 + \frac{1}{\lambda}) \right). \tag{4.2}$$

We now state a query complexity bound for Confident MC-LSPI combined with any UncertaintyCheck that satisfies certain assumptions.

Proposition 4.1. *Assume the* UncertaintyCheck *subroutine does not make any queries to the simulator. Assume when* UncertaintyCheck *returns a status of* uncertain *it also returns a tuple* $z = (z_s, z_a, z_\phi, z_q)$ *containing a feature* z_ϕ *that satisfies* $z_\phi^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} z_\phi > \tau$*, where* $\mathcal{C}$ *is the core set maintained by* Confident MC-LSPI *when* UncertaintyCheck *returns* uncertain*. Then the total number of queries to the simulator used by* Confident MC-LSPI *will be at most* $C_{max}^2 KnH$*.*

Proof. Notice that tuples are added to the core set only if UncertaintyCheck returns a status of uncertain (line 6 and line 14 in Confident MC-LSPI). Also, by assumption, when UncertaintyCheck returns uncertain then it only returns tuples $z = (z_s, z_a, z_\phi, z_q)$ containing features z_ϕ that satisfy $z_\phi^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} z_\phi > \tau$. Thus, we can use Lemma 4.1 to get that the core set size is bounded by $C_{\max}$. The total number of times policy iteration is restarted (restart means line 14 in Confident MC-LSPI is reached) is thus at most $C_{\max}$. Each run of policy iteration can take as much as K policy improvement steps. In each such step Rollout is run at most $C_{\max}$ times. Rollout does n rollouts of length H which queries the simulator once for each step (line in 10 in Rollout). UncertaintyCheck does not query the simulator at all. In total, the number of queries performed by Confident MC-LSPI is bounded by $C_{\max}^2 KnH$. □

We are now ready to state a query complexity bound for Confident MC-LSPI combined with UncertaintyCheck-Naive.

Proposition 4.2. *The total number of queries to the simulator used by* Confident MC-LSPI *combined with* UncertaintyCheck-Naive *can be bounded by* $C_{max}^2 KnH$*.*

When combined with the parameter settings for K, n, H *in Appendix A.1 we get that* $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{d^3}{\kappa^2(1-\gamma)^8}\big)$ *if* $\epsilon = 0$ *and* $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{d^2}{\epsilon^2(1-\gamma)^4}\big)$ *if* $\epsilon > 0$*, with the* $\tilde{\mathcal{O}}$ *notation hiding poly*$(\log(1/\delta), \log(1+b))$ *factors.*

Proof. Notice that when UncertaintyCheck-Naive returns uncertain it also only return tuples containing features $\phi(s,a)$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ that satisfy $\phi(s,a)^\top(\Phi^\top\Phi + \lambda I)^{-1}\phi(s,a) > \tau$ (lines 3-6 in UncertaintyCheck-Naive). Also, UncertaintyCheck-Naive does not make any queries to the simulator. Thus, we can apply Proposition 4.1 to get the $C_{\max}^2 KnH$ bound. The second part of the proposition can be shown by simple algebra after plugging in the parameter settings for K, n, H from Appendix A.1 into $C_{\max}^2 KnH$. □

4.3.2 Computational Complexity Bound

Proposition 4.3. *With parameter settings as defined in Appendix A.1. The computational complexity of* Confident MC-LSPI *combined with* UncertaintyCheck-Naive *can be bounded by poly*$(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}), |\mathcal{A}|)$ *if* $\epsilon = 0$ *and poly*$(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b), |\mathcal{A}|)$ *if* $\epsilon > 0$.

Proof. Notice the only loops not accounted for by the query complexity bound (Proposition 4.2) is the loop over all actions in UncertaintyCheck-Naive (line 2). This loop over all $a \in \mathcal{A}$ introduces the $|\mathcal{A}|$ dependence in the computation complexity. Further, the mathematical operations (line 16 in Confident MC-LSPI and line 3 in UncertaintyCheck-Naive) only require matrix multiplication and matrix inversion operations, which take at most $\text{poly}(C_{\max}, d)$ elementary artihmetic steps. Sampling from the policy (line 17 in Confident MC-LSPI) can be implemented with at most $|\mathcal{A}|$ computation by using a loop over all the actions to compute the arg max. Thus, we get the desired result by scaling the query complexity with an additional $|\mathcal{A}|$ and $\text{poly}(C_{\max}, d)$ factor and plugging in the parameter settings defined in Appendix A.1. □

4.3.3 Optimality of the Output Policy

Yin et al. (2021) used a *virtual algorithm* (VA) and *main algorithm* (MA) to prove the suboptimality of the 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). Assume UncertaintyCheck-Naive is used for the UncertaintyCheck subroutine in Confident MC-LSPI and Rollout throughout this subsection. 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. Fix an iteration $k \in [K]$ and a loop l with $\mathcal{C}$ the random core set during loop l. Notice that $\mathcal{C}$ only depends on the randomness from the previous $l-1$ loops. We condition on all the randomness from the previous $l-1$ loops, and thus $\mathcal{C}$ will be considered as a deterministic quantity now. Define $V_{\mathcal{C}} = \Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}} + \lambda I$ and a weighted matrix norm as $\|x\|_B^2 = x^\top B x,\ x \in \mathbb{R}^d, B \in \mathbb{R}^{d\times d}$. The VA's q-function at iteration k and loop l 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 $\mathcal{D} = \{\phi(s,a) : \|\phi(s,a)\|_{V_{\mathcal{C}}^{-1}}^2 \le \tau, (s,a) \in \mathcal{S}\times\mathcal{A}\}$ 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 by Yin et al. (2021).

Lemma 4.2 (Lemma B.2 of Yin et al. (2021))**.** *Suppose that Assumptions 1 and 2 holds and let $\theta > 0$. Fix $k \in [K]$. 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

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

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

$$\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty \leq \eta \tag{4.3}$$

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 the above result is useful, we state a classic policy improvement result, which can be found as Lemma B.3 of Yin et al. (2021) and in other papers.

Lemma 4.3 (Approximate policy iteration)**.** *Consider a sequence of policies $\pi_0, \pi_1, \pi_2, \cdots, \pi_K$ and a sequence of action-value functions $\tilde{q}_0, \ldots, \tilde{q}_{K-1}$. Suppose that for all $k = 1, 2, \cdots, K$, $\|\tilde{q}_{k-1} - q_{\pi_{k-1}}\|_\infty \leq \tilde{\eta}$, and π_k is greedy with respect to $\tilde{q}_{k-1}$. Then*

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

In our case the VA's policy $\tilde{\pi}_k$ is greedy w.r.t. $\tilde{q}_{k-1}$, and and thus via a union bound over all $k \in [K]$ and using Eq. (4.3) we have that with probability at least $1 - 2KC_{\max}\exp(-2\theta^2(1-\gamma)^2 n)$,

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

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

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

Case 2: $\|\phi(s,a)\|^2_{V_C^{-1}} \leq \tau$ 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 4.1), 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. Thus, in this loop policy iteration will never restart and line 19 in CONFIDENT MC-LSPI will be reached, causing for policy π_{K-1} to be returned. The loop during which line 19 in CONFIDENT MC-LSPI is reached we call the *final loop*. It turns out that if the MA and VA behave identically in the final loop, then it allows us to bound the suboptimality of the MA in the final loop, by using the result in Eq. (4.3) we have for the VA. More precisely, the following result, which we extracted from the work Yin et al. (2021) holds.

Proposition 4.4 (Equation (B.15) of Yin et al. (2021))**.** *Fix iteration $k \in [K]$, loop l and condition on the randomness from the previous $l-1$ loops. If $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty \leq \tilde{\eta}$ and $\max_{a \in \mathcal{A}} |\tilde{w}_k^\top \phi(\rho, a) - q_{\tilde{\pi}_{k-1}}(\rho, a)| \leq \tilde{\eta}$ and the VA and MA behave identically in the final loop, then with probability at least $1 - 4KC_{max}^2 \exp(-2\theta^2(1-\gamma)^2 n)$ we have*

$$v^*(\rho) - v_{\pi_{K-1}}(\rho) \leq \frac{8\tilde{\eta}}{(1-\gamma)^2} + \frac{2\gamma^{K-1}}{(1-\gamma)^2} \tag{4.4}$$

Notice, that we require three assumptions to be satisfied to use the above result: Firstly, we need a bound on $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty$. Secondly, we need a bound on $\max_{a \in \mathcal{A}} |\tilde{w}_k^\top \phi(\rho, a) - q_{\tilde{\pi}_{k-1}}(\rho, a)|$. Finally, we need to ensure that the VA and MA behave identically in the final loop. The next result shows that the suboptimality of

the MA's output policy π_{K-1} can be bounded, since (as we show in the proof) the three assumptions mentioned above are indeed satisfied.

Proposition 4.5. *Suppose Assumptions 1 and 2 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-LSPI *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.1) such that with probability at least $1-\delta$ the policy π_{K-1} returned by* Confident MC-LSPI *combined with* UncertaintyCheck-Naive *satisfies*

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

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-LSPI *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.1) such that with probability at least $1 - \delta$ the policy π_{K-1} satisfies*

$$v^*(\rho) - v_{\pi_{K-1}}(\rho) \leq \tfrac{64\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

Proof. Our goal is to use Proposition 4.4, so we should make sure that the three assumptions required by the proposition are satisfied. We have that $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty \leq \eta$ holds with probability at least $1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 n)$ by Eq. (4.3). We have that $|\tilde{w}_k^\top \phi(\rho, a) - q_{\tilde{\pi}_{k-1}}(\rho, a)| \leq \eta,\ \forall a \in \mathcal{A}$ holds with probability at least $1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 n)$ by Lemma 4.2. We can use Lemma 4.2 since we know that $\phi(\rho, a) \in \mathcal{D},\ \forall a \in \mathcal{A}$ by lines 3-8 in Confident MC-LSPI. UncertaintyCheck-Naive ensures that MA and VA behave identically in the final loop. It does 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-Naive 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. Thus, we have all the assumptions of Proposition 4.4 satisfied and can apply Proposition 4.4 with $\tilde{\eta} = \eta$ and the parameters set according to Appendix A.1 to get our desired suboptimality result. □

We are finally ready for to prove Theorem 1.

4.3.4 Proof of Theorem 1

The suboptimality bound follows by applying Proposition 4.5. The query complexity bound follows by applying Proposition 4.2. The computation complexity bound follows by applying Proposition 4.3.

Chapter 5

Confident Monte-Carlo Least-Squares Policy Iteration for Large Action Spaces

In this chapter we will extend the CONFIDENT MC-LSPI algorithm to the large action space setting by introducing two new UNCERTAINTYCHECK subroutines (UNCERTAINTYCHECK-EGSS and UNCERTAINTYCHECK-DAV). More precisely, we will show that when CONFIDENT MC-LSPI is combined with either UNCERTAINTYCHECK-EGSS or UNCERTAINTYCHECK-DAV the computational complexity no longer depends on poly($|\mathcal{A}|$), while still providing reasonable suboptimality guarantees of the output policy. This is in contrast to CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-NAIVE which has a computation complexity that depends on poly($|\mathcal{A}|$) (Section 4.3.2).

To achieve this we will need to make some assumptions. For UNCERTAINTYCHECK-EGSS (more details on this algorithm in the next section) we will 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.

For UncertaintyCheck-DAV (Section 5.4) we introduce an assumption on the features (Assumption 4) under which Assumption 3 will be satisfied.

5.1 Uncertainty Check using Efficient Good Set Search (EGSS)

In this section we will introduce the Uncertainty Check-Efficient Good Set Search (Uncertainty Check-EGSS) algorithm, which will rely on Assumption 3 holding.

Fix a state $s \in \mathcal{S}$. Recall that the *good set* (given $\mathcal{C}$) was defined as $\mathcal{D} = \{\phi(s, a) : \|\phi(s, a)\|^2_{V_\mathcal{C}^{-1}} \leq \tau, (s, a) \in \mathcal{S} \times \mathcal{A}\}$. The UncertaintyCheck-Naive algorithm either finds an action $a \in \mathcal{A}$ who's feature $\phi(s, a)$ is not in the good set $\phi(s, a) \notin \mathcal{D}$ i.e.

$$\phi(s, a)^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} \phi(s, a) = \|\phi(s, a)\|^2_{V_\mathcal{C}^{-1}} > \tau, \tag{5.1}$$

or just returns certain when no such action can be found. However, this requires iterating over all the actions in general, which is why UncertaintyCheck-Naive suffers poly($|\mathcal{A}|$) computation (Section 4.3.2). As such, UncertaintyCheck-EGSS aims to efficiently (without computational dependence on $|\mathcal{A}|$) approximate UncertaintyCheck-Naive. Next we show that with computation independent of $|\mathcal{A}|$, one can find an action vector $a \in \mathcal{A}$ that approximately maximizes

$\phi(s,a)^\top V_{\mathcal{C}}^{-1}\phi(s,a)$, with $V_{\mathcal{C}} = \Phi_{\mathcal{C}}^\top \Phi_{\mathcal{C}}$ as before. This will serve as the logic behind the UncertaintyCheck-EGSS algorithm design.

Lemma 5.1 (Efficient good set search)**.** *Let Assumption 3 hold. Then, there exist a computationally efficient procedure that makes* $2d$ *calls to the greedy oracle and which ensures that either*

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

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

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

Proof. 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

$$(v^*, a_{\max}) := \operatorname*{arg\,max}_{v \in \{\pm e_i\}_{i=1}^d, a \in \mathcal{A}} \langle Lv, \phi(s,a)\rangle \tag{5.2}$$

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 3 is satisfied. Hence, $(v^*, a_{\max})$ can be computed in $2d$ calls to the linear optimization oracle. 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 $V_{\mathcal{C}}^{-1} = LL^\top$.

The procedure will check whether $\zeta := \langle Lv^*, \phi(s, a_{\max})\rangle^2 > \tau$. If this holds, we claim that $\max_a \|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} > \tau$, otherwise $\max_a \|\phi(s,a)\|^2_{V_{\mathcal{C}}^{-1}} \leq d\tau$.

We start the proof of this by noticing that

$$\begin{aligned}\max_{a\in\mathcal{A}} \|L^\top\phi(s,a)\|_\infty^2 &= \max_{v\in\{\pm e_i\}_{i=1}^d} \max_{a\in\mathcal{A}} \langle v, L^\top\phi(s,a)\rangle^2 = \max_{v\in\{\pm e_i\}_{i=1}^d} \max_{a\in\mathcal{A}} \langle Lv, \phi(s,a)\rangle^2 \\ &= \langle Lv^*, \phi(s,a_{\max})\rangle^2 = \zeta\,. \end{aligned} \tag{5.3}$$

Now, recall the bidirectional 2-norm to ∞-norm inequality that states that for any

Algorithm 4 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**
  $\hat{a} \leftarrow \arg\max_{a \in \mathcal{A}} \phi(s,a)^\top L v$
  if $\left(\phi(s,\hat{a})^\top L v\right)^2 > \tau$ **then**
    result $\leftarrow (s, \hat{a}, \phi(s,\hat{a}), \text{NONE})$
    return UNCERTAIN, result
  end if
end for
return CERTAIN, NONE

vector $x \in \mathbb{R}^d$, $\frac{1}{d}\|x\|_2^2 \le \|x\|_\infty^2 \le \|x\|_2^2$. Then we have that

$$\begin{aligned}
\frac{1}{d}\max_{a\in\mathcal{A}} \|\phi(s,a)\|_{V_{\mathcal{C}}^{-1}}^2 &= \frac{1}{d}\max_{a\in\mathcal{A}} \phi(s,a)^\top V_{\mathcal{C}}^{-1}\phi(s,a) \\
&= \frac{1}{d}\max_{a\in\mathcal{A}} \phi(s,a)^\top L L^\top \phi(s,a) \\
&= \frac{1}{d}\max_{a\in\mathcal{A}} \|L^\top \phi(s,a)\|_2^2 \\
&\le \max_{a\in\mathcal{A}} \|L^\top \phi(s,a)\|_\infty^2 \\
&= \zeta && \text{(by Eq. (5.3))} \\
&= \|L^\top \phi(s,a_{\max})\|_\infty^2 && \text{(also by Eq. (5.3))} \\
&\le \|L^\top \phi(s,a_{\max})\|_2^2 \\
&\le \|\phi(s,a_{\max})\|_{V_{\mathcal{C}}^{-1}}^2 && (5.4)
\end{aligned}$$

Hence, by the above inequalities, if $\zeta > \tau$ then $\|\phi(s,a_{\max})\|_{V_{\mathcal{C}}^{-1}}^2 > \tau$ also holds, while if $\zeta \le \tau$ then $\max_{a\in\mathcal{A}} \|\phi(s,a)\|_{V_{\mathcal{C}}^{-1}}^2 \le d\tau$, completing the proof. □

UncertaintyCheck-EGSS presented as Algorithm 4 is an implementation of Eq. (5.2) together with the comparison of whether $\zeta = \langle Lv^*, \phi(s,a_{\max})\rangle^2$ is greater than τ. Thus, the runtime of UncertaintyCheck-EGSS is independent of $|\mathcal{A}|$, as stated in Lemma 5.1.

5.2 EGSS Theoretical Guarantees

Recall that by Lemma 5.1 if UncertaintyCheck-EGSS returns certain, then we only have the guarantee that $\max_a \|\phi(s,a)\|^2_{V_C^{-1}} \leq d\tau$ (which is weaker by a factor of d than the $\max_a \|\phi(s,a)\|^2_{V_C^{-1}} \leq \tau$ guarantee we had for UncertaintyCheck-Naive). A bound on the term $\|\phi(s,a)\|_{V_C^{-1}}$ is crucial for bounding the extrapolation error of our least square estimate w_k (Lemma 4.2, which also holds verbatim with τ replaced with $d\tau$). For the case with no misspecification ($\epsilon = 0$) this results in the query complexity increasing by a factor of d, while for case with misspecification ($\epsilon > 0$) the suboptimality of the returned policy increases by a factor of $\sqrt{d}$, which is similar to linear bandits, where multiple works have suffered an extra $\sqrt{d}$ in the regret for oracle-efficient methods (Dani et al., 2008; Agrawal and Goyal, 2013; Abeille and Lazaric, 2017). Importantly, this allows for the computation to be independent of $|\mathcal{A}|$, making this approach viable for large action spaces. The result that characterizes the performance of Confident MC-LSPI combined with UncertaintyCheck-EGSS is summarized in the next theorem.

Theorem 2 (Confident MC-LSPI EGSS Suboptimality)**.** *Suppose Assumptions 1 to 3 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0,1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-LSPI *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.2) such that with probability at least $1-\delta$ the policy π_{K-1} returned by* Confident MC-LSPI *combined with* UncertaintyCheck-EGSS *satisfies*

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

The query complexity is $\mathcal{O}\big(\frac{d^4}{\kappa^2(1-\gamma)^8}\big)$ and computation complexity is

$$\textit{poly}\left(d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right).$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0,1]$, there exist a setting of the parame-*

ters τ, λ, H, K, n *of* CONFIDENT MC-LSPI *based on* $\epsilon, \delta, \gamma, d, b$ *(shown in Appendix A.2) such that with probability at least* $1-\delta$ *the policy* π_{K-1} *satisfies*

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

The query complexity is $\mathcal{O}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$ *and computation complexity is*

$$poly\left(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log\left(\frac{1}{\delta}\right), \log(1+b)\right).$$

5.3 EGSS Analysis

In this section we prove Theorem 2. The proof will proceed in these steps:

1. In Section 5.3.1 we show a bound on the number of queries needed by CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-EGSS.
2. In Section 5.3.2 we show a bound on the computational complexity of CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-EGSS.
3. In Section 5.3.3 we show the policy π_{K-1} output by CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-EGSS is nearly optimal.
4. In Section 5.3.4 we combine the above three parts to prove Theorem 2.

5.3.1 Query Complexity Bound

Proposition 5.1. *The total number of queries to the simulator used by* CONFIDENT MC-LSPI *combined with* UNCERTAINTYCHECK-EGSS *can be bounded by* $C_{max}^2 KnH$.

When combined with the parameter settings for K, n, H *in Appendix A.2 we get that* $C_{max}^2 KnH = \tilde{\mathcal{O}}\left(\frac{d^4}{\kappa^2(1-\gamma)^8}\right)$ *if* $\epsilon = 0$ *and* $C_{max}^2 KnH = \tilde{\mathcal{O}}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$ *if* $\epsilon > 0$*, with the* $\tilde{\mathcal{O}}$ *notation hiding poly*$(\log(1/\delta), \log(1+b))$ *factors.*

Proof. Notice that when UNCERTAINTYCHECK-EGSS returns UNCERTAIN it also only returns tuples containing features $\phi(s,a)$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ that satisfy

$\|\phi(s,a)\|^2_{V_C^{-1}} \geq \|\phi(s,a)\|^2_\infty > \tau$ (lines 5-8 in UncertaintyCheck-EGSS). Also, UncertaintyCheck-EGSS does not make any queries to the simulator. Thus, we can apply the first part of Proposition 4.1 to get the $C^2_{\max}KnH$ bound. The second part of the proposition can be shown by simple algebra after plugging in the parameter settings for K, n, H from Appendix A.2 into $C^2_{\max}KnH$. □

5.3.2 Computational Complexity Bound

Proposition 5.2. *Suppose Assumption 3 is satisfied. With parameter settings as defined in Appendix A.2, the computational complexity of* Confident MC-LSPI *combined with* UncertaintyCheck-EGSS *can be bounded by poly*$(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ *if* $\epsilon = 0$ *and poly*$(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ *if* $\epsilon > 0$.

Proof. Notice the only loops not accounted for by the query complexity bound (Proposition 5.1) is the loop over all $v \in \{\pm e_l\}_{l=1}^d$ in UncertaintyCheck-EGSS (line 3). This loop over all $v \in \{\pm e_l\}_{l=1}^d$ only introduces a $2d$ dependence in the computation complexity. Further, the mathematical operations (line 16 in Confident MC-LSPI and line 4 in UncertaintyCheck-EGSS) only require matrix multiplication and matrix inversion operations that will only require poly$(C_{\max}, d)$ elementary arithmetic operational steps. One can sample from policy π_k (line 17 in Confident MC-LSPI) by simply outputting the result of $\arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})$. Under 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. $\mathcal{G}(w,\phi)$). Thus sampling from the policy in line 17 of Confident MC-LSPI can be implemented in constant time i.e. $\mathcal{O}(1)$. Scaling the query complexity with poly$(C_{\max}, d)$ (which are terms already in the query complexity bound) and plugging in the parameter settings defined in Appendix A.2 gives the desired result. □

5.3.3 Optimality of the Output Policy

Proposition 5.3. *Suppose Assumptions 1 to 3 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-LSPI *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.2) such that with probability at least $1 - \delta$ the policy π_{K-1} returned by* Confident MC-LSPI *combined with* UncertaintyCheck-EGSS *satisfies*

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

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-LSPI *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.2) such that with probability at least $1 - \delta$ the policy π_{K-1} satisfies*

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

Proof. Our goal will be to use Proposition 4.4, so we must show the three assumptions in the proposition are satisfied. We first show that the VA and MA behave identically in the final loop. Fix an iteration $k \in [K]$ and a loop l with $\mathcal{C}$ the random core set during loop l. Notice that $\mathcal{C}$ only depends on the randomness from the previous $l-1$ loops. We condition on all the randomness from the previous $l - 1$ loops, and thus $\mathcal{C}$ will be considered as a deterministic quantity now. Notice that UncertaintyCheck-EGSS provides a weaker guarantee than UncertaintyCheck-Naive, when the returned result is certain. Specifically, when UncertaintyCheck-EGSS returns a result of certain, then Lemma 5.1 guarantees that $\|\phi(s, a)\|^2_{V_{\mathcal{C}}^{-1}} \leq d\tau$ for all $a \in \mathcal{A}$. While when the UncertaintyCheck-Naive returns a result of certain, then $\|\phi(s, a)\|^2_{V_{\mathcal{C}}^{-1}} \leq \tau$ for all $a \in \mathcal{A}$. Thus, we define a larger good set $\mathcal{D}_d = \{\phi(s, a) : \|\phi(s, a)\|^2_{V_{\mathcal{C}}^{-1}} \leq d\tau\}$.

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 larger good set $\mathcal{D}_d$. Thus, the VA's policy $\tilde{\pi}_k$ 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 $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty$ with this new definition of $\tilde{q}_{k-1}$. First we state a slight modification of Lemma 4.2 that holds for the larger good set $\mathcal{D}_d$

Lemma 5.2 (EGSS modified Lemma B.2 from Yin et al. (2021))**.** *Suppose that Assumption 1 holds 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

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

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-EGSS as discussed above. □

Essentially we get an extra $\sqrt{d}$ factor due to the larger 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 5.4 (approximate value function bound for EGSS)**.** *Suppose that Assumption 1 holds 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

$$|\tilde{q}_{k-1}(s,a) - q_{\tilde{\pi}_{k-1}}(s,a)| \leq \eta_2 \tag{5.5}$$

by Proposition 5.4. While for any $(s,a) \in (\mathcal{S} \times \mathcal{A})$ such that $\phi(s,a) \notin \mathcal{D}_d$, we have

$$|\tilde{q}_{k-1}(s,a) - q_{\tilde{\pi}_{k-1}}(s,a)| = |q_{\tilde{\pi}_{k-1}}(s,a) - q_{\tilde{\pi}_{k-1}}(s,a)| = 0 \tag{5.6}$$

□

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 5.2 to bound $\eta_2 \geq |\tilde{w}_k^\top \phi(\rho,a) - q_{\tilde{\pi}_{k-1}}(\rho,a)|,\ \forall a \in \mathcal{A}$. Thus, we have all the assumptions of Proposition 4.4 satisfied and can apply Proposition 4.4 with $\tilde{\eta} = \eta_2$ and the parameters set according to Appendix A.2 to get our desired suboptimality result. □

We are finally ready for to prove Theorem 2.

5.3.4 Proof of Theorem 2

The suboptimality bound follows by applying Proposition 5.3. The query complexity bound follows by applying Proposition 5.1. The computation complexity bound follows by applying Proposition 5.2.

5.4 Uncertainty Check using a Default Action Vector (DAV)

The result in Section 5.2 makes no restriction on the choice of features as long as the greedy policy can be computed efficiently (Assumptions 1 and 3) but it leaves open whether and when this can be done. In this section we address this by introducing an additive feature model for which the oracle can be implemented efficiently.

Assumption 4. *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* $m \geq 1$ *(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* $\phi(s, a^{(1:m)}) = \sum_{i=1}^{m} \phi_i(s, a^{(i)})$.

We will use the notation $a^{(1:m)} := (a^{(1)}, ..., a^{(m)}) \in \mathcal{A}^{(1:m)}$ with $a^{(i)} \in \mathcal{A}^{(i)}$ for all $i \in [m]$ and call $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ an *action vector* and one of its indices $a^{(i)},\ i \in [m]$ an action. The above assumption immediately gives rise to a corollary stating that for any policy π, the q_π-function is (approximately) linear in the feature map ϕ and decomposes additively across the components $\mathcal{A}^{(i)}$.

Corollary 5.1. *Suppose Assumptions 1 and 4 hold. Then, for each policy* π *there exists a weight vector* $w_\pi \in \mathbb{R}^d, \|w_\pi\|_2 \leq b$ *satisfying* $\max_{(s,a^{(1:m)}) \in \mathcal{S} \times \mathcal{A}^{(1:m)}} |q_\pi(s, a^{(1:m)}) - w_\pi^\top \sum_{j=1}^{m} \phi_j(s, a^{(j)})| \leq \epsilon$.

With the greedy oracle (Assumption 3), one can use Confident MC-LSPI combined with UncertaintyCheck-EGSS and directly invoke Theorem 2. However, in this section we will introduce a new uncertainty check algorithm, Uncertainty Check-Default Action Vector (UncertaintyCheck-DAV) as Algorithm 5, that explicitly uses the additive structure. The additive feature structure leads to improved results (compared to UncertaintyCheck-EGSS) in the regimes where the square root of the dimension is larger than m (discussed in Section 5.5). The additive model will also allow an efficient implementation of Confident MC-Politex from

Algorithm 5 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 $\leftarrow (s, \tilde{a}, \phi(s,\tilde{a}), \text{NONE})$
      return UNCERTAIN, result
    end if
  end for
end for
return CERTAIN, NONE

Yin et al. (2021), which leads to an improved dependence on the suboptimality in the misspecified setting (Chapter 7).

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:

$$\begin{aligned} &\arg\max_{a^{(1:m)} \in \mathcal{A}} w^\top \phi(s, a^{(1:m)}) \\ &= \big(\arg\max_{a^{(1)} \in \mathcal{A}^{(1)}} w^\top \phi_1(s, a^{(1)}), ..., \arg\max_{a^{(m)} \in \mathcal{A}^{(m)}} w^\top \phi_m(s, a^{(m)}) \big) \end{aligned} \tag{5.7}$$

A simple example when Assumptions 1 and 4 hold is when m agents "live" in m separate MDPs such that in each MDP the q-functions are linearly realizable (Assumption 4 is satisfied in each MDP) 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 Chapter 8 we show that Assumption 4 also captures MDPs that require cooperation between agents, and provide some empirical results.

The UncertaintyCheck-DAV algorithm only iterates over $\sum_{i=1}^{m} |A^{(i)}|$ action vectors instead of all the action vectors like UncertaintyCheck-Naive does. This of course achieves the goal of compute independent of $|\mathcal{A}^{(1:m)}| = \prod_{i=1}^{m} |A^{(i)}|$. Notice that $\sum_{i=1}^{m} |A^{(i)}|$ is significantly smaller than $\prod_{i=1}^{m} |A^{(i)}|$ as long as $|\mathcal{A}^{(i)}| > 1,\ \forall i \in [m]$ and m is not very small. The fact that Confident MC-LSPI combined with UncertaintyCheck-DAV still provides viable suboptimality guarantees (Theorem 3) is proved in Section 5.6.

5.5 DAV Theoretical Guarantees

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

Theorem 3 (Confident MC-LSPI DAV Suboptimality)**.** *Suppose Assumptions 1, 2 and 4 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-LSPI *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.3) such that with probability at least $1 - \delta$ the policy π_{K-1} returned by* Confident MC-LSPI *combined with* UncertaintyCheck-DAV *satisfies*

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

The query complexity is $\mathcal{O}\left(\frac{m^2 d^3}{\kappa^2 (1-\gamma)^8}\right)$ and computation complexity is

$$poly\left(\sum_{i=1}^{m} |\mathcal{A}^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right).$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-LSPI *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.3) such that with probability at least $1 - \delta$ the policy π_{K-1} satisfies*

$$v^*(\rho) - v_{\pi_{K-1}}(\rho) \leq \tfrac{128\epsilon\sqrt{dm}}{(1-\gamma)^2} (1 + \log(1 + b^2 \epsilon^{-2} d^{-1}))^{1/2}.$$

The query complexity is $\mathcal{O}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$ *and computation complexity is*

$$poly\left(\sum_{i=1}^{m}|\mathcal{A}^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log\left(\frac{1}{\delta}\right), \log(1+b)\right).$$

When compared to Theorem 1 where UncertaintyCheck-Naive is used we have an extra factor of m^2 in the query complexity for $\epsilon = 0$, while for $\epsilon > 0$ we only have an extra factor of m in the suboptimality of the output policy. On the other hand, the computational complexity is improved from $\mathcal{O}(\prod_{i=1}^{m}|\mathcal{A}^{(i)}|)$ to $\mathcal{O}(\sum_{i=1}^{m}|\mathcal{A}^{(i)}|)$. When compared to Theorem 2 where UncertaintyCheck-EGSS was used instead of UncertaintyCheck-DAV the extra dependence on $\sqrt{d}$ changed to m.

5.6 DAV Analysis

In this section we prove Theorem 3. The proof will proceed in these steps:

1. In Section 5.6.1 we show a bound on the number of queries needed by Confident MC-LSPI combined with UncertaintyCheck-DAV.
2. In Section 5.6.2 we show a bound on the computational complexity of Confident MC-LSPI combined with UncertaintyCheck-DAV.
3. In Section 5.6.3 we show the policy π_{K-1} output by Confident MC-LSPI combined with UncertaintyCheck-DAV is nearly optimal.
4. In Section 5.6.4 we combine the above three parts to prove Theorem 3.

5.6.1 Query Complexity Bound

Proposition 5.5. *The total number of queries to the simulator used by* Confident MC-LSPI *combined with* UncertaintyCheck-DAV *can be bounded by* $C_{max}^2 KnH$.

When combined with the parameter settings for K, n, H *in Appendix A.2 we get that* $C_{max}^2 KnH = \tilde{\mathcal{O}}\left(\frac{m^2 d^3}{\kappa^2(1-\gamma)^8}\right)$ *if* $\epsilon = 0$ *and* $C_{max}^2 KnH = \tilde{\mathcal{O}}\left(\frac{d^2}{\epsilon^2(1-\gamma)^4}\right)$ *if* $\epsilon > 0$, *with the* $\tilde{\mathcal{O}}$ *notation hiding poly*$(\log(1/\delta), \log(1+b))$ *factors.*

Proof. Notice that when UncertaintyCheck-DAV returns uncertain it also only returns tuples containing features $\phi(s,a)$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ that satisfy $\|\phi(s,a)\|^2_{V_\mathcal{C}^{-1}} \geq \|\phi(s,a)\|^2_\infty > \tau$ (lines 6-9 in UncertaintyCheck-DAV). Also, UncertaintyCheck-DAV does not make any queries to the simulator. Thus, we can apply the first part of Proposition 4.1 to get the $C^2_{\max} KnH$ bound. The second part of the proposition can be shown by simple algebra after plugging in the parameter settings for K, n, H from Appendix A.3 into $C^2_{\max} KnH$. □

5.6.2 Computational Complexity Bound

Proposition 5.6. *Suppose Assumption 4 is satisfied. With parameter settings as defined in Appendix A.3. The computational complexity of* Confident MC-LSPI *combined with* UncertaintyCheck-DAV *can be bounded by*

$$poly\left(\sum_{i=1}^m |A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right)$$

if $\epsilon = 0$ *and*

$$poly\left(\sum_{i=1}^m |A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\epsilon}, \log\left(\tfrac{1}{\delta}\right), \log(1+b)\right)$$

if $\epsilon > 0$.

Proof. Notice the only loops not accounted for by the query complexity bound (Proposition 5.5) is the loop over all $j \in [m]$ (line 3 in UncertaintyCheck-DAV) and the loop over all $a^{(j)} \in \mathcal{A}^{(j)}$ (line 4 in UncertaintyCheck-DAV). These loops only introduce a $\sum_{i=1}^m |\mathcal{A}^{(i)}|$ dependence in the computation complexity. Further, the mathematical operations (line 16 in Confident MC-LSPI and line 6 in UncertaintyCheck-DAV) only require matrix multiplication and matrix inversion operations, which will only require poly$(C_{\max}, d)$ computation. One can sample from policy π_k (line 17 in Confident MC-LSPI) by simply outputting the result of $\arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})$. Under Assumption 4 $\arg\max_{\tilde{a}\in\mathcal{A}} w^\top \phi(s,\tilde{a})$ can be computed in $\mathcal{O}(d\sum_{i=1}^m |\mathcal{A}^{(i)}|)$ time as shown in Eq. (5.7). Thus, sampling from the policy in line

17 of CONFIDENT MC-LSPI can be implemented with $\mathcal{O}(d\sum_{i=1}^{m}|\mathcal{A}^{(i)}|)$ computation time. Scaling the query complexity with poly$(\sum_{i=1}^{m}|\mathcal{A}^{(i)}|, C_{\max}, d)$ and plugging in the parameter settings defined in Appendix A.3 gives the desired result. □

5.6.3 Optimality of the Output Policy

Proposition 5.7. *Suppose Assumptions 1, 2 and 4 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-LSPI *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.3) such that with probability at least $1 - \delta$ the policy π_{K-1} returned by* CONFIDENT MC-LSPI *combined with* UNCERTAINTYCHECK-DAV *satisfies*

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

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-LSPI *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.3) such that with probability at least $1 - \delta$ the policy π_{K-1} satisfies*

$$v^*(\rho) - v_{\pi_{K-1}}(\rho) \leq \tfrac{128\epsilon\sqrt{dm}}{(1-\gamma)^2}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

Proof. Recall that CONFIDENT MC-LSPI sets a *default action vector* $\bar{a}^{(1:m)} \in \mathcal{A}^{(1:m)}$ as a global. Define a subset of $\mathcal{A}^{(1:m)}$ 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 after changing the action at index i in $\bar{a}^{(1:m)}$ with $a^{(i)}$. 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 $\|\phi(s, a^{(1:m)})\|^2_{V_{\mathcal{C}}^{-1}} > \tau$.

Our goal will be to use Proposition 4.4, so we must show the three assumptions in the proposition are satisfied. Fix an iteration $k \in [K]$ and a loop l with $\mathcal{C}$ the random core set during loop l. Notice that $\mathcal{C}$ only depends on the randomness from

the previous $l-1$ loops. We condition on all the randomness from the previous $l-1$ loops, and thus $\mathcal{C}$ will be considered as a deterministic quantity now. We show 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. Thus we are sure that $\phi(s, a^{(1:m)}) \in \mathcal{D}$ for all $a^{(1:m)} \in \bar{\mathcal{A}}^{(1:m)}$ and for all the states encountered in the final loop. Which means that the states encountered in the final loop must be in $\bar{\mathcal{S}}$. Thus, the VA's policy $\tilde{\pi}_k$ 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 $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty$. First we state a slight modification of Lemma 4.2 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 5.3 (Modification of Lemma B.2 in (Yin et al., 2021))**.** *Suppose that Assumption 4 holds, 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 5.8 (approximate value function bound for DAV)**.** *Suppose that Assumption 4 holds, 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 $(s, a^{(1:m)}) \in (\bar{\mathcal{S}} \times \mathcal{A}^{(1:m)})$, with probability at least $1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 n)$, we have

$$\begin{aligned}
&|\tilde{q}_{k-1}(s, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&= |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&= |\tilde{w}_k^\top \phi(s, a^{(1:m)}) \pm w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&\leq |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)})| + |w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| \\
&\leq |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)})| + \epsilon \\
&= |\tilde{w}_k^\top \phi(s, a^{(1:m)}) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, a^{(1:m)}) + (m-1)\tilde{w}_k^\top \phi(s, \bar{a}^{(1:m)}) - (m-1)\tilde{w}_k^\top \phi(s, \bar{a}^{(1:m)}) \\
&\quad + (m-1)w_{\tilde{\pi}_{k-1}}^\top \phi(s, \bar{a}^{(1:m)}) - (m-1)w_{\tilde{\pi}_{k-1}}^\top \phi(s, \bar{a}^{(1:m)})| + \epsilon \\
&= \left| \sum_{i=1}^{m} \tilde{w}_k^\top \phi(s, (a^{(i)}, \bar{a}^{(-i)})) - w_{\tilde{\pi}_{k-1}}^\top \phi(s, (a^{(i)}, \bar{a}^{(-i)})) \right| \qquad (5.8) \\
&\quad + \left| (m-1)\left[w_{\tilde{\pi}_{k-1}}^\top \phi(s, \bar{a}^{(1:m)})) - \tilde{w}_k^\top \phi(s, \bar{a}^{(1:m)} \right] \right| + \epsilon \\
&\leq m\bar{\eta} + (m-1)\bar{\eta} + \epsilon \\
&= \bar{\eta}(2m-1) + \epsilon \qquad (5.9)
\end{aligned}$$

where the second last inequality holds by Lemma 5.3 (because the features of all the state action pairs considered are in $\mathcal{D}$, since $s \in \bar{\mathcal{S}}$).

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

$$|\tilde{q}_{k-1}(s, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| = |q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(s, a^{(1:m)})| = 0\,. \qquad (5.10)$$

□

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. (5.9) to bound $\eta_1 \geq |\tilde{w}_k^\top \phi(\rho, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(\rho, a^{(1:m)})|, \ \forall a^{(1:m)} \in \mathcal{A}^{(1:m)}$.

Thus, we have all the assumptions of Proposition 4.4 satisfied and can apply Proposition 4.4 with $\tilde{\eta} = \eta_1$ and the parameters set according to Appendix A.3 to get our desired suboptimality result. □

We are finally ready for to prove Theorem 3.

5.6.4 Proof of Theorem 3

The suboptimality bound follows by applying Proposition 5.7. The query complexity bound follows by applying Proposition 5.5. The computation complexity bound follows by applying Proposition 5.6.

Chapter 6

Confident Monte-Carlo Politex

Similar to how Chapter 4 was devoted to introducing the Confident MC-LSPI algorithm presented by Yin et al. (2021), we will devote this chapter to introduce the Confident MC-Politex algorithm also presented by Yin et al. (2021). The Confident MC-Politex algorithm can be found as Algorithm 6 and when combined with UncertaintyCheck-Naive we (almost[1]) have the Confident MC-Politex algorithm that is presented by Yin et al. (2021).

The Politex algorithm has been shown to obtain better suboptimality gaurantees than LSPI (when $\epsilon > 0$) by Abbasi-Yadkori et al. (2019), and the same holds for Confident MC-Politex when compared to Confident MC-LSPI (Yin et al., 2021). In particular, the suboptimality of Confident MC-Politex combined with UncertaintyCheck-Naive scales as $\mathcal{O}(\sqrt{d}\epsilon/(1-\gamma))$, which is also known to be unavoidable when the computation is kept polynomial Weisz et al. (2022). The trade-off however, is that Confident MC-Politex has a larger query complexity than Confident MC-LSPI.

Although Confident MC-Politex is also based on policy iteration (like Confident MC-LSPI) an important difference is that it uses stochastic policies based

[1]We do not clip our action value-functions to the $[0, (1-\gamma)^{-1}]$ interval in line 17 of Confident MC-Politex, while Yin et al. (2021) does.

Algorithm 6 CONFIDENT MC-POLITEX

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, $\forall z \in \mathcal{C}$ ▷ 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: $\quad \pi_k(a|s) \leftarrow \exp\left(\alpha \sum_{j=1}^k w_j^\top \phi(s,a)\right) / \sum_{a' \in \mathcal{A}} \exp\left(\alpha \sum_{j=1}^k w_j^\top \phi(s,a')\right).$
18: **end for**
19: **return** $\bar{\pi}_{K-1} \sim \text{Unif}\{\pi_k\}_{k=0}^{K-1}$

on an exponential weighting of each actions action-value (line 17 in CONFIDENT MC-POLITEX). Besides this difference, the final policy $\bar{\pi}_{K-1}$ that is returned by CONFIDENT MC-POLITEX is a uniform sample from all policies that are created during the policy improvement phase, namely $\pi_0, \cdots, \pi_{K-1}$.

An important difference to note between the CONFIDENT MC-POLITEX algorithm presented here and the CONFIDENT MC-POLITEX algorithm presented in Yin et al. (2021) is that we do no *clip* the q-function (in line 17). In particular, the policy used in CONFIDENT MC-POLITEX by Yin et al. (2021) is the following

$$\pi_k(a|s) \leftarrow \exp\left(\alpha \sum_{j=0}^{k-1} q_j(s,a)\right) / \sum_{a \in \mathcal{A}} \exp\left(\alpha \sum_{j=0}^{k-1} q_k(s,a)\right) \tag{6.1}$$

With $q_{k-1}(s,a) = \min\{\max\{w_k^\top \phi(s,a), 0\}, 1/(1-\gamma)\}$. Notice that the q-function q_j here is clipped such that it is in the $[0, (1-\gamma)^{-1}]$ interval. The policy we use for

CONFIDENT MC-POLITEX (line 17) is the following

$$\pi_k(a|s) \leftarrow \exp\left(\alpha \sum_{j=1}^{k} w_j^\top \phi(s,a)\right) / \sum_{a' \in \mathcal{A}} \exp\left(\alpha \sum_{j=1}^{k} w_j^\top \phi(s,a')\right).$$

Notice that we do not clip the q-function and use $w_j^\top \phi$ directly. This means that we no longer have the guarantee that $w_j^\top \phi$ maps to $[0, (1-\gamma)^{-1}]$. The purpose of removing the clipping of the q-function in the version of CONFIDENT MC-POLITEX that we present here is so that we can efficiently sample from the policy when Assumption 4 is satisfied. How this can be done will be explained in Section 7.1. Importantly, removing the clipping does not suffer any increase in the dominating terms of the final policies suboptimality (shown in Theorem 4).

6.1 Theoretical Guarantees

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

Theorem 4 (CONFIDENT MC-POLITEX Suboptimality (Theorem 5.2 in Yin et al. (2021))). *Suppose Assumptions 1 and 2 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0,1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-POLITEX *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.4) such that with probability at least $1-\delta$ the policy $\bar{\pi}_{K-1}$ returned by* CONFIDENT MC-POLITEX *combined with* UNCERTAINTYCHECK-NAIVE *satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \kappa$$

The query complexity is $\mathcal{O}\left(\frac{d^3}{\kappa^4(1-\gamma)^9}\right)$ and computation complexity is

$$\text{poly}\left(|\mathcal{A}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right).$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0,1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-POLITEX *based on $\epsilon, \delta, \gamma, d, b$ (shown in*

Appendix A.4) such that with probability at least $1-\delta$ *the policy* $\bar{\pi}_{K-1}$ *satisfies*

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

The query complexity is $\mathcal{O}\left(\frac{d}{\epsilon^4(1-\gamma)^5}\right)$ *and computation complexity is*

$$poly\left(|\mathcal{A}|, d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log\left(\frac{1}{\delta}\right), \log(1+b)\right).$$

As discussed earlier, when $\epsilon > 0$, the suboptimality is better (scales with $1/(1-\gamma)$) than that of Confident MC-LSPI (Theorem 3), which scales with $1/(1-\gamma)^2$. However, the query complexity is worse (as is typical for Politex).

6.2 Analysis

In this section we prove Theorem 4. Similar to the analysis section for Confident MC-LSPI (Section 4.3), in this section the proof is borrowed directly from Yin et al. (2021) with only a few minor adjustments, since as mentioned earlier we do not clip our value functions. The proof will proceed in these steps:

1. In Section 6.2.1 we show a bound on the number of queries needed by Confident MC-Politex combined with UncertaintyCheck-Naive.
2. In Section 6.2.2 we show a bound on the computational complexity of Confident MC-Politex combined with UncertaintyCheck-Naive.
3. In Section 6.2.3 we show the policy $\bar{\pi}_{K-1}$ output by Confident MC-Politex combined with UncertaintyCheck-Naive is nearly optimal.
4. In Section 6.2.4 we combine the above three parts to prove Theorem 4.

6.2.1 Query Complexity Bound

We first state a query complexity bound for Confident MC-Politex combined with any UncertaintyCheck that satisfies certain assumptions.

Proposition 6.1. *Assume the* UncertaintyCheck *subroutine does not make any queries to the simulator. Assume when* UncertaintyCheck *returns a status of* uncertain *it also only returns tuples* $z = (z_s, z_a, z_\phi, z_q)$ *containing features* z_ϕ *that satisfy* $z_\phi^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} z_\phi > \tau$*, where* $\mathcal{C}$ *is the core set maintained by* Confident MC-Politex *when* UncertaintyCheck *returns* uncertain.*. Then the total number of queries to the simulator used by* Confident MC-Politex *can then be bounded by* $C_{max}^2 KnH$*.*

Proof. This proof is nearly identical to the proof of Proposition 4.1, except with Confident MC-LSPI replaced with Confident MC-Politex here. Notice that tuples are added to the core set only if UncertaintyCheck returns a status of uncertain (line 6 and line 14 in Confident MC-Politex). Also, by assumption, when UncertaintyCheck returns uncertain then it only returns tuples $z = (z_s, z_a, z_\phi, z_q)$ containing features z_ϕ that satisfy $z_\phi^\top (\Phi_\mathcal{C}^\top \Phi_\mathcal{C} + \lambda I)^{-1} z_\phi > \tau$. Thus, we can use Lemma 4.1 to get that the core set size is bounded by $C_{\max}$. The total number of times Policy iteration is restarted (restart means line 14 in 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 (line in 10 in Rollout). UncertaintyCheck does not query the simulator at all. In total, the number of queries performed by Confident MC-Politex is bounded by $C_{\max}^2 KnH$. □

We are now ready to state a query complexity bound for Confident MC-Politex combined with UncertaintyCheck-Naive.

Proposition 6.2. *The total number of queries to the simulator used by* Confident MC-Politex *combined with* UncertaintyCheck-Naive *can be bounded by* $C_{max}^2 KnH$*.*

When combined with the parameter settings for K, n, H in Appendix A.1 we get that $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{d^3}{\kappa^4(1-\gamma)^9}\big)$ if $\epsilon = 0$ and $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{d}{\epsilon^4(1-\gamma)^5}\big)$ if $\epsilon > 0$, with the $\tilde{\mathcal{O}}$ notation hiding poly$(\log(1/\delta), \log(1+b))$ factors.

Proof. Notice that when UncertaintyCheck-Naive returns uncertain it also only returns tuples containing features $\phi(s,a)$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ that satisfy $\phi(s,a)^\top (\Phi^\top \Phi + \lambda I)^{-1} \phi(s,a) > \tau$ (lines 3-6 in UncertaintyCheck-Naive). Also, UncertaintyCheck-Naive does not make any queries to the simulator. Thus, we can apply Proposition 6.1 to get the $C_{\max}^2 KnH$ bound. The second part of the proposition can be shown by simple algebra after plugging in the parameter settings for K, n, H from Appendix A.4 into $C_{\max}^2 KnH$. □

6.2.2 Computational Complexity Bound

Proposition 6.3. *With parameter settings as defined in Appendix A.4. The computational complexity of* Confident MC-Politex *combined with* UncertaintyCheck-Naive *can be bounded by poly$(d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}), |\mathcal{A}|)$ if $\epsilon = 0$ and poly$(d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b), |\mathcal{A}|)$ if $\epsilon > 0$.*

Proof. Notice the only loops not accounted for by the query complexity bound (Proposition 6.2) is the loop over all actions in UncertaintyCheck-Naive (line 2). This loop over all $a \in \mathcal{A}$ introduces a $|\mathcal{A}|$ dependence in the computation complexity. Further, the mathematical operations (line 16 in Confident MC-Politex and line 3 in UncertaintyCheck-Naive) only require matrix multiplication and matrix inversion operations, which will only require poly$(C_{\max}, d)$ computation. Sampling from the policy (line 17 in Confident MC-Politex) can be implemented with at most $K|\mathcal{A}|$ computation by using a loop over all the actions $\mathcal{A}$ and policy iterations K. Thus, we get the desired result by scaling the query complexity with an additional $K|\mathcal{A}|$ and poly$(C_{\max}, d)$ factor and plugging in the parameter settings defined in Appendix A.4. □

6.2.3 Optimality of the Output Policy

It turns out the story for Confident MC-Politex is extremely similar to that of Confident MC-LSPI 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 4.3 (since it relied on a greedy policy), and thus cannot use Proposition 4.4 to bound the suboptimality 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.

Fix an iteration $k \in [K]$ and a loop l with $\mathcal{C}$ the random core set during loop l. Notice that $\mathcal{C}$ only depends on the randomness from the previous $l-1$ loops. We condition on all the randomness from the previous $l-1$ loops, and thus $\mathcal{C}$ will be considered as a deterministic quantity now. Recall that we do not clip the q-function estimates in our Confident MC-Politex algorithm. This means we must define the VA's q-function differently from the way it was defined for Confident MC-Politex in Yin et al. (2021), by removing clipping for the case when $\phi(s, a^{(1:m)}) \in \mathcal{D}$. This leads to the exact same definition of the VA's q-function as we have already seen in Section 4.3.3, which we restate here

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

We define the VA's policy as

$$\tilde{\pi}_k(a|s) \propto \exp\left(\alpha \sum_{j=0}^{k-1} \tilde{q}_j(s,a)\right). \tag{6.2}$$

Recall that since the reward function r is assumed to be in $[0,1]$ we know that for any policy π that $q_\pi \in [0, (1-\gamma)^{-1}]$. Since we do not use clipping $\tilde{q}_k,\ k \in [K]$ may not be in the $[0, (1-\gamma)^{-1}]$ interval. However, if we have a bound $\tilde{\eta}$ on $\|\tilde{q}_k - q_{\tilde{\pi}_k}\|_\infty$, then we can be sure that $\tilde{q}_k \in [-\tilde{\eta}, (1-\gamma)^{-1} + \tilde{\eta}]$. 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. We replace the interval $[0, (1-\gamma)^{-1}]$ (in Lemma D.1 from Yin et al. (2021)) with the interval $[-\tilde{\eta}, (1-\gamma)^{-1}+\tilde{\eta}]$, since as discussed above $\tilde{q}_k \in [-\tilde{\eta}, (1-\gamma)^{-1}+\tilde{\eta}]$.

Lemma 6.1 (modified Lemma D.1 in Yin et al. (2021) also in Szepesvári (2022b))**.** *Given an initial policy π_0, a sequence of functions $q_k : \mathcal{S} \times \mathcal{A} \to [-\tilde{\eta}, (1-\gamma)^{-1}+\tilde{\eta}]$, $k \in [K-1]$, construct a sequence of policies $\pi_1, ..., \pi_{K-1}$ according to Eq.* (6.2) *with $\alpha = ((1-\gamma)^{-1}+2\tilde{\eta})\sqrt{\frac{2\log(|\mathcal{A}|)}{K}}$. If $q_{\pi_k} \in [0, 1/(1-\gamma)]$ and $\tilde{\eta} \geq \|\tilde{q}_k - q_{\pi_k}\|_\infty$, $\forall k \in [K-1]$. Then, for any $s \in \mathcal{S}$, the mixture policy $\bar{\pi}_{K-1} \sim Unif\{\pi_k\}_{k=0}^{K-1}$ satisfies*

$$v^*(s) - v_{\bar{\pi}_{K-1}}(s) \leq \left(\frac{1}{(1-\gamma)^2} + \frac{2\tilde{\eta}}{1-\gamma}\right)\sqrt{\frac{2\log(|\mathcal{A}|)}{K}} + \frac{2\tilde{\eta}}{1-\gamma} \tag{6.3}$$

Notice that the above result suggests we just need to control the term $\|q_k - q_{\pi_k}\|_\infty$ for all $k \in [K-1]$. Using Lemma 6.1 (instead of Lemma D.1 in Yin et al. (2021)), one can extract another slightly modified result from Yin et al. (2021).

Proposition 6.4 (equation (D.8) in Yin et al. (2021))**.** *Fix iteration $k \in [K]$, loop l and condition on the randomness from the previous $l-1$ loops. If $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty \leq \tilde{\eta}$ and $\max_{a\in\mathcal{A}} |\tilde{w}_k^\top \phi(\rho, a) - q_{\tilde{\pi}_{k-1}}(\rho, a)| \leq \tilde{\eta}$ and the VA and MA behave identically in the final loop, then with probability at least $1 - 4KC_{max}^2 \exp(-2\theta^2(1-\gamma)^2 n)$ we have*

$$v^*(s) - v_{\bar{\pi}_{K-1}}(\rho) \leq \left(\frac{1}{(1-\gamma)^2} + \frac{2\tilde{\eta}}{1-\gamma}\right)\sqrt{\frac{2\log(|\mathcal{A}|)}{K}} + \frac{4\tilde{\eta}}{1-\gamma} \tag{6.4}$$

Notice, that we require the same three things as in the CONFIDENT MC-LSPI case (Proposition 4.4). We need a bound on $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty$. We need a bound on $\max_{a\in\mathcal{A}} |\tilde{w}_k^\top \phi(\rho, a) - q_{\tilde{\pi}_{k-1}}(\rho, a)|_\infty$. We need to ensure that the VA and MA behave identically in the final loop. Then, we can get a bound on the suboptimality of the MA's output policy $\bar{\pi}_{K-1}$.

The next result shows that the suboptimality of the MA's output policy $\bar{\pi}_{K-1}$ can be bounded, since (as we show in the proof) the three assumptions mentioned above are indeed satisfied.

Proposition 6.5. *Suppose Assumptions 1 and 2 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-Politex *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.4) such that with probability at least $1-\delta$ the policy $\bar{\pi}_{K-1}$ returned by* Confident MC-Politex *combined with* UncertaintyCheck-Naive *satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \kappa$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-Politex *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.4) such that with probability at least $1 - \delta$ the policy $\bar{\pi}_{K-1}$ satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \tfrac{64\epsilon\sqrt{d}}{1-\gamma}(1 + \log(1 + b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

Proof. Our goal is use Proposition 6.4, so we should make sure that the three assumptions required by the proposition are satisfied. Recall from Section 4.3, we have that $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty \leq \eta$ holds with probability at least $1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 n)$ by Eq. (4.3). We also have that $|\tilde{w}_k^\top \phi(\rho, a) - q_{\tilde{\pi}_{k-1}}(\rho, a)| \leq \eta,\ \forall a \in \mathcal{A}$ holds with probability at least $1 - 2C_{\max}\exp(-2\theta^2(1-\gamma)^2 n)$ by Lemma 4.2. We can use Lemma 4.2 since we know that $\phi(\rho, a) \in \mathcal{D},\ \forall a \in \mathcal{A}$ by lines 3-8 in Confident MC-Politex. UncertaintyCheck-Naive ensures that MA and VA behave identically in the final loop. It does 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-Naive 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. Thus, we have all the assumptions of Proposition 6.4 are

satisfied and can apply Proposition 6.4 with $\tilde{\eta} = \eta$ and the parameters set according to Appendix A.4 to get our desired suboptimality result. □

6.2.4 Proof of Theorem 4

The suboptimality bound follows by applying Proposition 6.5. The query complexity bound follows by applying Proposition 6.2. The computation complexity bound follows by applying Proposition 6.3.

Chapter 7

Confident Monte-Carlo Politex for Large Action Spaces

In this chapter we will extend the CONFIDENT MC-POLITEX algorithm to the large action space setting by using our previously introduced UNCERTAINTYCHECK algorithms (UNCERTAINTYCHECK-EGSS and UNCERTAINTYCHECK-DAV). More precisely, we will show that when CONFIDENT MC-POLITEX is combined with either UNCERTAINTYCHECK-EGSS or UNCERTAINTYCHECK-DAV the computational complexity no longer depends on poly($|\mathcal{A}|$), while still providing reasonable suboptimality guarantees of the output policy. This is in contrast to CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-NAIVE which has computation complexity that depends on poly($|\mathcal{A}|$) (Section 6.2.2). In Chapter 6 we discussed how CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-NAIVE obtains a better suboptimality but worse query complexity when compared to CONFIDENT MC-LSPI. In this chapter we will see that a similar story occurs when CONFIDENT MC-POLITEX combined with either UNCERTAINTYCHECK-EGSS or UNCERTAINTYCHECK-DAV is compared to CONFIDENT MC-LSPI. Namely the suboptimality of CONFIDENT MC-POLITEX is better than that of CONFIDENT MC-LSPI (by a factor of $(1-\gamma)^{-1}$), while for the query complexity it is worse.

7.1 Confident MC-Politex Without Value Function Clipping

In this section we describe why it was necessary to remove clipping of the q-functions in order to sample from the Confident MC-Politex policy with computation that does not depend on poly($|\mathcal{A}|$). Recall that the Confident MC-LSPI policy is

$$\pi_k(a|s) \leftarrow \mathbb{1}\left(a = \arg\max_{\tilde{a}\in\mathcal{A}} w_k^\top \phi(s,\tilde{a})\right). \tag{7.1}$$

Thus, we could use Assumption 3 directly to compute the arg max in constant time and sample from the policy. Further, when Assumption 4 holds we showed that this leads to Assumption 3 being satisfied and the argmax oracle being implementable with $\sum_{i=1}^m |A^{(i)}|d$ computation (Eq. (5.7)).

The policy used in Confident MC-Politex by Yin et al. (2021) is the following

$$\pi_k(a|s) \leftarrow \exp\left(\alpha \sum_{j=0}^{k-1} q_j(s,a)\right) / \sum_{a\in\mathcal{A}} \exp\left(\alpha \sum_{j=0}^{k-1} q_k(s,a)\right) \tag{7.2}$$

With $q_{k-1}(s,a) = \min\{\max\{w_k^\top \phi(s,a), 0\}, 1/(1-\gamma)\}$. Notice that the q-function q_j here is clipped such that it is in the $[0,(1-\gamma)^{-1}]$ interval. It is not immediate how to make use of Assumption 3 to sample from the Confident MC-Politex policy, with clipping, since there is no arg max that needs to be computed. Further, we are not aware of an efficient way to compute the clipped q-function for all action-vectors in $\mathcal{A}$ (i.e. $\exp(\alpha \sum_{j=0}^{k-1} q_j(s,a))$ for all $a \in \mathcal{A}$) if only Assumption 3 or even Assumption 4 is satisfied. However, if we remove the clipping, then if Assumption 4 is satisfied we will show that, indeed, it is possible to compute the Confident MC-Politex policy efficiently. Importantly, removing the clipping does not suffer any increase in the dominating terms of the output policies suboptimality (shown in Theorem 4). Recall the Confident MC-Politex policy without clipping is defined as follows (line 17 in Confident MC-Politex)

$$\pi_k(a|s) \leftarrow \exp\left(\alpha \sum_{j=1}^{k} w_j^\top \phi(s,a)\right) / \sum_{a'\in\mathcal{A}} \exp\left(\alpha \sum_{j=1}^{k} w_j^\top \phi(s,a')\right) \tag{7.3}$$

We now show the above policy can be sampled from efficiently.

Proposition 7.1 (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$(\sum_{i=1}^m |A^{(i)}|, K, d)$.

Proof. Fix arbitrary $a^{(1:m)} \in \mathcal{A}^{(1:m)}$. To sample from π_k it is sufficient to sample action vectors $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ proportional to $\exp(\alpha \sum_{j=0}^{k-1} w_j^\top \phi(s, a^{(1:m)}))$. Rearranging $\exp(\alpha \sum_{j=0}^{k-1} w_j^\top \phi(s, a^{(1:m)}))$ and plugging in that $\phi(s, a^{(1:m)}) = \sum_{i=1}^m \phi_i(s, a^{(i)})$ under Assumption 4 we have

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

Which means that the probability of sampling an action vector $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ is equal to the product of the probabilities of sampling $a^{(i)} \in \mathcal{A}^{(i)}$ for $i \in [m]$ independently. Notice that if we can compute $\prod_{j=0}^{k-1} \exp(\alpha w_j^\top \phi_i(s, a^{(i)}))$ for all $a^{(i)} \in \mathcal{A}^{(i)}$, then can sample $a^{(i)} \in \mathcal{A}^{(i)}$ proportional to $\prod_{j=0}^{k-1} \exp(\alpha w_j^\top \phi_i(s, a^{(i)}))$ by ordering the actions $a^{(i)} \in \mathcal{A}^{(i)}$, then sampling a random number x uniformly in the range $[0, \sum_{a^{(i)} \in \mathcal{A}^{(i)}} \prod_{j=0}^{k-1} \exp(\alpha w_j^\top \phi_i(s, a^{(i)}))]$ and selecting the first action for which the partial sum over $\prod_{j=0}^{k-1} \exp(\alpha w_j^\top \phi_i(s, a^{(i)}))$ for the ordered actions is greater than x. For $i \in [m]$, sampling an action $a^{(i)} \in \mathcal{A}^{(i)}$ means computing $\prod_{j=0}^{k-1} \exp(\alpha w_j^\top \phi_i(s, a^{(i)}))$ for all $a^{(i)} \in \mathcal{A}^{(i)}$. This requires at most poly$(|\mathcal{A}^{(i)}|, K, d)$ computation. Thus, we can sample an action vector $a^{(1:m)} \in \mathcal{A}^{(1:m)}$ with computation poly$(\sum_{i=1}^m |A^{(i)}|, K, d)$

by sampling one action $a^{(i)} \in \mathcal{A}^{(i)}$ from each index $i \in [m]$ and then combining the actions into an action vector $a^{(1:m)} = (a^{(1)}, \ldots, a^{(m)}) \in \mathcal{A}^{(1:m)}$. □

Since by Proposition 7.1 to sample from the policy in Confident MC-Politex efficiently we need Assumption 4 to hold, we will assume that it does hold for the remainder of this chapter. Recall that when Assumption 4 holds this leads to Assumption 3 being satisfied and the argmax oracle implementable with $\sum_{i=1}^{m} |A^{(i)}|d$ computation (Eq. (5.7)). Thus, we can still use the UncertaintyCheck-EGSS algorithm if Assumption 4 holds. In the next sections we provide guarantees on the performance of Confident MC-Politex combined with UncertaintyCheck-EGSS or UncertaintyCheck-DAV.

7.2 EGSS Theoretical Guarantees

The result that characterizes the performance of Confident MC-Politex combined with UncertaintyCheck-EGSS is summarized in the next theorem.

Theorem 5 (Confident MC-Politex EGSS Suboptimality)**.** *Suppose Assumptions 1, 2 and 4 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-Politex *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.5) such that with probability at least $1-\delta$ the policy $\bar{\pi}_{K-1}$ returned by* Confident MC-Politex *combined with* UncertaintyCheck-EGSS *satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \kappa$$

The query complexity is $\mathcal{O}\left(\frac{md^4}{\kappa^4(1-\gamma)^9}\right)$ and computation complexity is

$$poly\left(\sum_{i=1}^{m} |A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right).$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parame-*

ters τ, λ, H, K, n *of* CONFIDENT MC-POLITEX *based on* $\epsilon, \delta, \gamma, d, b$ *(shown in Appendix A.5) such that with probability at least* $1-\delta$ *the policy* $\bar{\pi}_{K-1}$ *satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \tfrac{64\epsilon d}{1-\gamma}(1+\log(1+b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

The query complexity is $\mathcal{O}\left(\frac{md}{\epsilon^4(1-\gamma)^5}\right)$ *and computation complexity is*

$$poly\left(\sum_{i=1}^{m}|A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\epsilon}, \log\left(\tfrac{1}{\delta}\right), \log(1+b)\right).$$

When compared to CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-NAIVE (Theorem 1) we have an extra factor of d in the query complexity for $\epsilon = 0$, while for $\epsilon > 0$ we only have an extra factor of $\sqrt{d}$ in the suboptimality of the output policy. This is expected, and the reason for this has been in discussed at the start of Section 5.2. Importantly, we have no dependence on $|\mathcal{A}| = \prod_{i=1}^{m}|A^{(i)}|$, making this approach viable for large action spaces. Unlike in the CONFIDENT MC-LSPI case, when $\epsilon > 0$ the suboptimality only scales with $(1-\gamma)^{-1}$, while it scaled with $(1-\gamma)^{-2}$ for CONFIDENT MC-LSPI (which is better for CONFIDENT MC-POLITEX). However, the query complexity for CONFIDENT MC-POLITEX scales with an extra factor of $m\epsilon^{-2}(1-\gamma)^{-1}$. Unsurprisingly, this is nearly the same pattern (without the m factor in the query complexity) as we saw when comparing CONFIDENT MC-POLITEX to CONFIDENT MC-LSPI combined with UNCERTAINTYCHECK-NAIVE in Section 6.1.

7.3 EGSS Analysis

In this section we prove Theorem 5. We mirror the same proof structure and borrow many of the proof techniques as in Section 5.3, since as we have discussed earlier, CONFIDENT MC-POLITEX only differs from CONFIDENT MC-LSPI by the policy it uses (line 17). However, we still explicitly write all of the proof steps for completeness. The proof will proceed in these steps:

1. In Section 7.3.1 we show a bound on the number of queries needed by CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-EGSS.

2. In Section 7.3.2 we show a bound on the computational complexity of CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-EGSS.

3. In Section 7.3.3 we show the policy $\bar{\pi}_{K-1}$ output by CONFIDENT MC-POLITEX combined with UNCERTAINTYCHECK-EGSS is nearly optimal.

4. In Section 7.3.4 we combine the above three parts to prove Theorem 5.

7.3.1 Query Complexity Bound

Proposition 7.2. *The total number of queries to the simulator used by* CONFIDENT MC-POLITEX *combined with* UNCERTAINTYCHECK-EGSS *can be bounded by* $C_{max}^2 KnH$.

When combined with the parameter settings for K, n, H in Appendix A.5 we get that $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{md^4}{\kappa^4(1-\gamma)^9}\big)$ if $\epsilon = 0$ and $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{md}{\epsilon^4(1-\gamma)^5}\big)$ if $\epsilon > 0$, with the $\tilde{\mathcal{O}}$ notation hiding poly$(\log(1/\delta), \log(1+b))$ factors.

Proof. Notice that when UNCERTAINTYCHECK-EGSS returns UNCERTAIN it also only returns tuples containing features $\phi(s,a)$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ that satisfy $\|\phi(s,a)\|^2_{V_C^{-1}} \geq \|\phi(s,a)\|^2_\infty > \tau$ (lines 5-8 in UNCERTAINTYCHECK-EGSS). Also, UNCERTAINTYCHECK-EGSS does not make any queries to the simulator. Thus, we can apply the first part of Proposition 6.1 to get the $C_{\max}^2 KnH$ bound. The second part of the proposition can be shown by simple algebra after plugging in the parameter settings for K, n, H from Appendix A.5 into $C_{\max}^2 KnH$. □

7.3.2 Computational Complexity Bound

Proposition 7.3. *Suppose Assumption 4 is satisfied. With parameter settings as defined in Appendix A.5. The computational complexity of* CONFIDENT MC-

POLITEX *combined with* UNCERTAINTYCHECK-EGSS *can be bounded by*

$$poly\left(\sum_{i=1}^{m}|A^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\kappa}, \log\left(\frac{1}{\delta}\right)\right)$$

if $\epsilon = 0$ *and*

$$poly\left(\sum_{i=1}^{m}|A^{(i)}|, d, \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log\left(\frac{1}{\delta}\right), \log(1+b)\right)$$

if $\epsilon > 0$.

Proof. Notice the only loops not accounted for by the query complexity bound (Proposition 5.1) is the loop over all $v \in \{\pm e_l\}_{l=1}^{d}$ in UNCERTAINTYCHECK-EGSS (line 3). This loop over all $v \in \{\pm e_l\}_{l=1}^{d}$ only introduces a $2d$ dependence in the computation complexity. Further, the mathematical operations (line 16 in CONFIDENT MC-POLITEX and line 4 in UNCERTAINTYCHECK-EGSS) only require matrix multiplication and matrix inversion operations, which will only require $\text{poly}(C_{\max}, d)$ computation. Since Assumption 4 is satisfied, we can sample from policy π_k (line 17 in CONFIDENT MC-POLITEX) with $\text{poly}(\sum_{i=1}^{m}|A^{(i)}|, K, d)$ computation (Proposition 7.1). Scaling the query complexity with $\text{poly}(\sum_{i=1}^{m}|A^{(i)}|, K, C_{\max}, d)$ and plugging in the parameter settings defined in Appendix A.5 gives the desired result. □

7.3.3 Optimality of the Output Policy

Proposition 7.4. *Suppose Assumptions 1, 2 and 4 hold.*

1. *Assume that* $\epsilon = 0$. *Then, for any* $\kappa > 0, \delta \in (0, 1]$, *there exist a setting of the parameters* τ, λ, H, K, n *of* CONFIDENT MC-POLITEX *based on* $\kappa, \delta, \gamma, d, b$ *(shown in Appendix A.5) such that with probability at least* $1-\delta$ *the policy* $\bar{\pi}_{K-1}$ *returned by* CONFIDENT MC-POLITEX *combined with* UNCERTAINTYCHECK-EGSS *satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \kappa$$

2. *Assume* $\epsilon > 0$. *Then, for any* $\delta \in (0, 1]$, *there exist a setting of the parameters* τ, λ, H, K, n *of* CONFIDENT MC-POLITEX *based on* $\epsilon, \delta, \gamma, d, b$ *(shown in*

Appendix A.5) such that with probability at least $1-\delta$ the policy $\bar{\pi}_{K-1}$ satisfies

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \tfrac{64\epsilon d}{1-\gamma}(1+\log(1+b^2\epsilon^{-2}d^{-1}))^{1/2}.$$

Proof. Our goal will be to use Proposition 6.4, so we must show the three assumptions in the proposition are satisfied. We first show that the VA and MA behave identically in the final loop. We follow the same proof technique as Proposition 5.3.

Define the VA's q-function at iteration k in the same way as in Section 5.3.3

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

and the VA's policy as

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

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 larger good set $\mathcal{D}_d$. Thus, the VA's policy $\tilde{\pi}_k$ would always be based on $\tilde{q}_{k-1} = \tilde{w}_k^\top \phi$ in the final loop. This ensures that the VA and MA behave identically in the final loop.

We also know $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty \leq \eta_2$ by Proposition 5.4.

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-Politex 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 5.2 to bound $\eta_2 \geq |\tilde{w}_k^\top \phi(\rho, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(\rho, a^{(1:m)})|,\ \forall a^{(1:m)} \in \mathcal{A}^{(1:m)}$. Thus, we have all the assumptions of Proposition 6.4 satisfied and can apply Proposition 6.4 with $\tilde{\eta} = \eta_2$ and the parameters set according to Appendix A.5 to get our desired suboptimality result. □

We are finally ready for to prove Theorem 5.

7.3.4 Proof of Theorem 5

The suboptimality bound follows by applying Proposition 7.4. The query complexity bound follows by applying Proposition 7.2. The computation complexity bound follows by applying Proposition 7.3.

7.4 DAV Theoretical Guarantees

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

Theorem 6 (CONFIDENT MC-POLITEX DAV Suboptimality)**.** *Suppose Assumptions 1, 2 and 4 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-POLITEX *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.6) such that with probability at least $1-\delta$ the policy $\bar{\pi}_{K-1}$ returned by* CONFIDENT MC-POLITEX *combined with* UNCERTAINTYCHECK-DAV *satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \kappa$$

The query complexity is $\mathcal{O}\left(\frac{m^3 d^3}{\kappa^4 (1-\gamma)^9}\right)$ and computation complexity is

$$poly\left(\sum_{i=1}^{m} |A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right).$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* CONFIDENT MC-POLITEX *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.6) such that with probability at least $1 - \delta$ the policy $\bar{\pi}_{K-1}$ satisfies*

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

The query complexity is $\mathcal{O}\left(\frac{md}{\epsilon^4 (1-\gamma)^5}\right)$ and computation complexity

$$poly\left(\sum_{i=1}^{m} |A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\epsilon}, \log\left(\tfrac{1}{\delta}\right), \log(1+b)\right).$$

As we have already seen even when UncertaintyCheck-Naive is used (Theorem 4) the suboptimality is better for Confident MC-Politex (scales with $1/(1-\gamma)$) than that of Confident MC-LSPI (Theorem 1), which scales with $1/(1-\gamma)^2$. However, the query complexity is worse. Importantly, we have avoided a dependence on $|\mathcal{A}^{(1:m)}| = \prod_{i=1}^{m} |A^{(i)}|$ in the computation complexity by using UncertaintyCheck-DAV instead of UncertaintyCheck-Naive. However, the trade-off is that we have introduced an extra factor of m^3 in the query complexity when $\epsilon = 0$, and a extra factor of m in the suboptimality when $\epsilon > 0$.

7.5 DAV Analysis

In this section we prove Theorem 6. Similar to Section 7.3 we borrow many of the proof techniques from the corresponding Confident MC-LSPI section (Section 5.6). The proof will proceed in these steps:

1. In Section 7.5.1 we show a bound on the number of queries needed by Confident MC-Politex combined with UncertaintyCheck-DAV.
2. In Section 7.5.2 we show a bound on the computational complexity of Confident MC-Politex combined with UncertaintyCheck-DAV.
3. In Section 7.5.3 we show the policy $\bar{\pi}_{K-1}$ output by Confident MC-Politex combined with UncertaintyCheck-DAV is nearly optimal.
4. In Section 7.5.4 we combine the above three parts to prove Theorem 6.

7.5.1 Query Complexity Bound

Proposition 7.5. *The total number of queries to the simulator used by* Confident MC-Politex *combined with* UncertaintyCheck-DAV *can be bounded by* $C_{max}^2 KnH$.

When combined with the parameter settings for K, n, H in Appendix A.5 we get

that $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{m^3 d^3}{\kappa^4 (1-\gamma)^9}\big)$ *if* $\epsilon = 0$ *and* $C_{max}^2 KnH = \tilde{\mathcal{O}}\big(\frac{md}{\epsilon^4 (1-\gamma)^5}\big)$ *if* $\epsilon > 0$*, with the* $\tilde{\mathcal{O}}$ *notation hiding poly*$(\log(1/\delta), \log(1+b))$ *factors.*

Proof. Notice that when UncertaintyCheck-DAV returns uncertain it also only returns tuples containing features $\phi(s,a)$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ that satisfy $\|\phi(s,a)\|^2_{V_\mathcal{C}^{-1}} \geq \|\phi(s,a)\|^2_\infty > \tau$ (lines 6-9 in UncertaintyCheck-DAV). Also, UncertaintyCheck-DAV does not make any queries to the simulator. Thus, we can apply the first part of Proposition 6.1 to get the $C^2_{\max} KnH$ bound. The second part of the proposition can be shown by simple algebra after plugging in the parameter settings for K, n, H from Appendix A.6 into $C^2_{\max} KnH$. □

7.5.2 Computational Complexity Bound

Proposition 7.6. *Suppose Assumption 4 is satisfied. With parameter settings as defined in Appendix A.6. The computational complexity of* Confident MC-Politex *combined with* UncertaintyCheck-DAV *can be bounded by*

$$poly\left(\sum_{i=1}^m |A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\kappa}, \log\left(\tfrac{1}{\delta}\right)\right)$$

if $\epsilon = 0$ *and*

$$poly\left(\sum_{i=1}^m |A^{(i)}|, d, \tfrac{1}{1-\gamma}, \tfrac{1}{\epsilon}, \log\left(\tfrac{1}{\delta}\right), \log(1+b)\right)$$

if $\epsilon > 0$.

Proof. Notice the only loops not accounted for by the query complexity bound (Proposition 7.5) is the loop over all $j \in [m]$ (line 3 in UncertaintyCheck-DAV) and the loop over all $a^{(j)} \in \mathcal{A}^{(j)}$ (line 4 in UncertaintyCheck-DAV). These loops only introduce a $\sum_{i=1}^m |A^{(i)}|$ dependence in the computation complexity. Further, the mathematical operations (line 16 in Confident MC-Politex and line 6 in UncertaintyCheck-DAV) only require matrix multiplication and matrix inversion operations, which will only require poly$(C_{\max}, d)$ computation. Since Assumption 4 is satisfied, we can sample from policy π_k (line 17 in Confident MC-Politex) with

poly($\sum_{i=1}^m |A^{(i)}|, K, d$) computation (Proposition 7.1). Scaling the query complexity with poly($\sum_{i=1}^m |A^{(i)}|, C_{\max}, Kd$) and plugging in the parameter settings defined in Appendix A.6 gives the desired result. □

7.5.3 Optimality of the Output Policy

Proposition 7.7. *Suppose Assumptions 1, 2 and 4 hold.*

1. *Assume that $\epsilon = 0$. Then, for any $\kappa > 0, \delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-Politex *based on $\kappa, \delta, \gamma, d, b$ (shown in Appendix A.6) such that with probability at least $1-\delta$ the policy $\bar{\pi}_{K-1}$ returned by* Confident MC-Politex *combined with* UncertaintyCheck-DAV *satisfies*

$$v^*(\rho) - v_{\bar{\pi}_{K-1}}(\rho) \leq \kappa$$

2. *Assume $\epsilon > 0$. Then, for any $\delta \in (0, 1]$, there exist a setting of the parameters τ, λ, H, K, n of* Confident MC-Politex *based on $\epsilon, \delta, \gamma, d, b$ (shown in Appendix A.6) such that with probability at least $1 - \delta$ the policy $\bar{\pi}_{K-1}$ satisfies*

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

Proof. Our goal will be to use Proposition 6.4, so we must show the three assumptions in the proposition are satisfied. We first show that the VA and MA behave identically in the final loop. Define the VA's q-function at iteration k in the same way as in Section 5.6.3

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

and the VA's policy as

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

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.

Thus we are sure that $\phi(s, a^{(1:m)}) \in \mathcal{D}$ for all $a^{(1:m)} \in \bar{\mathcal{A}}^{(1:m)}$ and for all the states encountered in the final loop. Which means that the states encountered in the final loop must be in $\bar{\mathcal{S}}$. Thus, the VA's policy $\tilde{\pi}_k$ would always be based on $\tilde{q}_{k-1} = \tilde{w}_k^\top \phi$ in the final loop. This ensures that the VA and MA behave identically in the final loop.

We also know $\|\tilde{q}_{k-1} - q_{\tilde{\pi}_{k-1}}\|_\infty \leq \eta_1$ by Proposition 5.8.

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-POLITEX 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. (5.9) to bound $\eta_1 \geq |\tilde{w}_k^\top \phi(\rho, a^{(1:m)}) - q_{\tilde{\pi}_{k-1}}(\rho, a^{(1:m)})|, \ \forall a^{(1:m)} \in \mathcal{A}^{(1:m)}$.

Thus, we have all the assumptions of Proposition 6.4 satisfied and can apply Proposition 6.4 with $\tilde{\eta} = \eta_1$ and the parameters set according to Appendix A.6 to get our desired suboptimality result. □

We are finally ready for to prove Theorem 6.

7.5.4 Proof of Theorem 6

The suboptimality bound follows by applying Proposition 7.7. The query complexity bound follows by applying Proposition 7.5. The computation complexity bound follows by applying Proposition 7.6.

Chapter 8

Examples and Experiments

This chapter is structured as follows. In Section 8.1 we provide some experiments to show the effectiveness of our approaches. Specifically, we run our algorithms on a product MDP with 4 agents. In Section 8.2 we show that Assumptions 1 and 4 (which are needed by UncertaintyCheck-DAV (Sections 5.5 and 7.4)) are satisfied by more than just product MDPs. In particular, we show that there exists an MDP, that is not a product MDP, under which Assumptions 1 and 4 are satisfied. The MDP we use requires each agent to consider the policy of other agents in order to pick the action that maximizes the action-value function of the joint policy.

8.1 Experimental Results

We evaluate the performance of the proposed algorithms on a product MDP (composed of grid worlds) shown in Fig. 8.1. Each of the 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

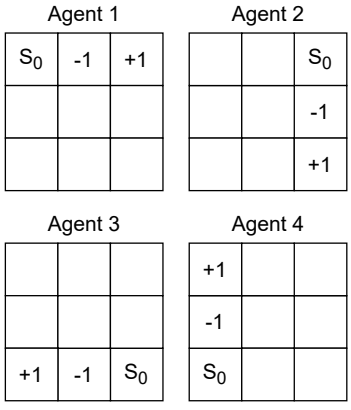

Figure 8.1: Four agent grid world.

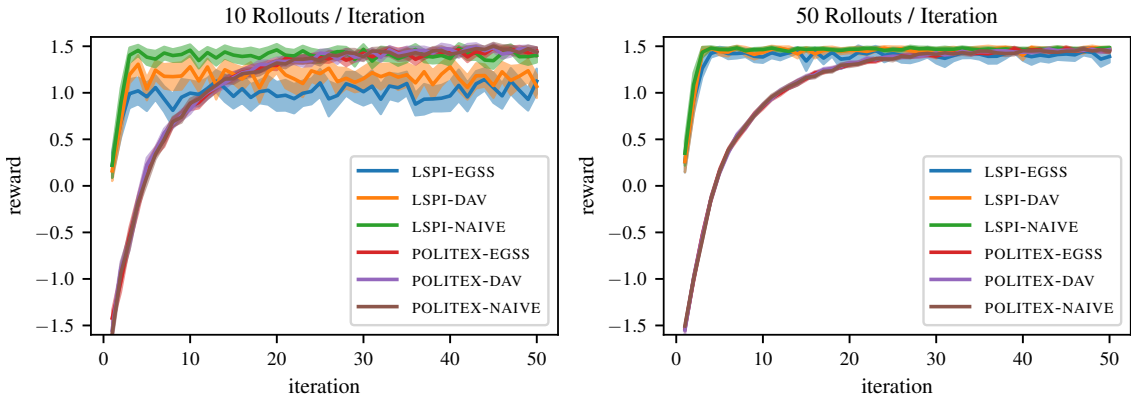

Figure 8.2: Numerical results on a grid world with four agents.

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.

We run each variant of the algorithm for 50 iterations ($K = 50$) without resets (in line 14 of CONFIDENT MC-LSPI we no longer **goto** line $(*)$). The reason for running the experiments without resets is because they run faster. When we compared to the case with resets (for 10 random seeds) the performance of the algorithms was very similar, thus we did not use resets for the experiments presented here. 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 $d = 4 \cdot 9 \cdot 4 = 144$. One can easily verify that these features satisfy Assumption 1 with $\epsilon = 0$. 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 8.2 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 com-

bined with UncertaintyCheck-EGSS and UncertaintyCheck-DAV. In addition we compare to UncertaintyCheck-Naive which iterates over all $|A| = 4^4 = 256$ actions. The intention of the experiments is to show that when Confident MC-LSPI or Confident MC-Politex is combined with either UncertaintyCheck-EGSS or UncertaintyCheck-DAV the resulting value of the output policy is not much worse than when UncertaintyCheck-Naive is used. Important is that UncertaintyCheck-EGSS or UncertaintyCheck-DAV have computation independent of $|\mathcal{A}|$, which means they run faster than UncertaintyCheck-Naive, and would run much faster if $|\mathcal{A}|$ was larger than in this toy example.

With only 10 rollouts (left plot in Fig. 8.2), the policy of Confident MC-LSPI (for all 3 UncertaintyCheck subroutines) after iteration 5 does not show any signs of improving, is noisy, and is 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 can sometimes leads to a deterioration in the policy improvement step. Further, the policy of UncertaintyCheck-Naive gets higher reward than UncertaintyCheck-DAV which gets higher reward than UncertaintyCheck-EGSS. One way this behaviour can be explained is by looking at our theoretical results[1] for UncertaintyCheck-DAV and UncertaintyCheck-EGSS in Section 5.5 and Section 5.2 respectively. When compared to the theoretical result for UncertaintyCheck-Naive in Section 4.2 we can see for the $\epsilon = 0$ case the query complexity of UncertaintyCheck-DAV has an extra m^2 factor, while UncertaintyCheck-EGSS has an extra d factor. Also, notice that the query complexity bound for all 3 UncertaintyCheck subroutines has a $1/\kappa^2$ term, where $\kappa > 0$ was the level of suboptimality you would like to guarantee. Since $C_{\max}, K, H, n$ are held constant in our experiments, the number of queries used by all 3 Uncer-

[1]Keeping in mind that the theoretical results are for the worst case MDP, which likely is not the case here. However, one would hope the results give some insight for other (not worst case) instances.

taintyCheck subroutines should be approximately the same. As such, we can see that $1/\kappa^2_{\text{naive}} = m^2/\kappa^2_{\text{dav}} = d/\kappa^2_{\text{egss}}$ should hold since the number of queries used by all 3 UncertaintyCheck subroutines should be approximately the same. This gives us that $\kappa_{\text{dav}} = m\kappa_{\text{naive}}$, and $\kappa_{\text{egss}} = \sqrt{d}\kappa_{\text{naive}}$, which implies that we should suspect UncertaintyCheck-DAV and UncertaintyCheck-EGSS to perform worse than UncertaintyCheck-Naive. Further, since $\sqrt{d} = 12 > 4 = m$, we should also suspect that UncertaintyCheck-EGSS will perform worse than UncertaintyCheck-DAV. This is exactly what we observe in the left plot of Fig. 8.2.

Now focusing on the Confident MC-Politex case, still for 10 rollouts per iteration. Using the same arguments as above for comparing query complexity of the 3 UncertaintyCheck subroutines we end up with $1/\kappa^4_{\text{naive}} = m^3/\kappa^4_{\text{dav}} = md/\kappa^4_{\text{egss}}$ needing to hold. Where we used $1/\kappa^4$ instead of $1/\kappa^2$ now, since the query complexity bound for all 3 UncertaintyCheck subroutines has a $1/\kappa^4$ term for the Confident MC-Politex algorithm (Sections 6.1, 7.2 and 7.4). This implies $\kappa_{\text{dav}} = m^{3/4}\kappa_{\text{naive}}$, and $\kappa_{\text{egss}} = (md)^{1/4}\kappa_{\text{naive}}$, which have smaller multiples in front of $1/\kappa_{\text{naive}}$ than for Confident MC-LSPI and may explain why the difference between the 3 UncertaintyCheck subroutines is not as large for Confident MC-Politex as it was for Confident MC-LSPI. Also, Confident MC-Politex is much more stable, but also requires more iterations to converge. This is expected because in Confident MC-Politex the policy estimates from all iterations are averaged.

With 50 rollouts per iteration (right plot in Fig. 8.2), the same observations as for the 10 rollouts per iteration case still hold. However, all the curves have become smoother and the error bars have reduced, which is expected since the number rollouts controls how well our parameter estimate w_k concentrates (Lemma 4.2).

8.2 Cooperation Example

In this section we show that there exists an MDP, that is not a product MDP, and requires some notion of cooperation between agents (i.e. the best action for one agent depends on the policy of another agent). Consider the MDP in Fig. 8.3, which can be verified to satisfy Assumption 1 with $\gamma = 1/2$ (shown in the next subsection). The feature map is defined in Section 8.2.1. We will assume that Assumption 4 holds with $m = 2$. This means we have 2 agents in the MDP and their action sets are $\mathcal{A}^{(1)} = \mathcal{A}^{(2)} = \{0, 1\}$. The joint action set is $\mathcal{A}^{(1:2)} = \mathcal{A}^{(1)} \times \mathcal{A}^{(2)} = \{(0,0), (1,0), (0,1), (1,1)\}$. The starting state is $\rho = s_1$ and any action vector $a^{(1:2)} \in \mathcal{A}^{(1:2)}$ taken in s_1 causes the agents to move to s_2 or s_3, which are absorbing states (i.e. the agents will remain in s_2 or s_3 once they get there). In s_1 the action taken by agent 1 alone determines the transition of both the agents (i.e. if $a^{(1)} \in \mathcal{A}^{(1)}$ is 0 then the agents transition to state s_3, while if $a^{(1)} = 1$ then the agents transition to state s_2). If the agents are in s_2 or s_3, then the action of agent 2 alone determines the reward (i.e. if the agents are in s_2 and $a^{(2)} \in \mathcal{A}^{(2)}$ is 1 then the agents get a reward of 1, while if the agents are in s_3 and $a^{(2)} = 0$ then the agents get a reward of 1, and the agents get a reward of 0 otherwise). If we fix the policy for one of the agents, then the other agent will face a reduced MDP where the transitions and rewards only depend on its actions. We will show that the best action for the first agent in s_1 will be different depending on how the second agent acts (through the second agents policy), which suggests that the first agent should cooperate with the second agent to achieve a higher value. It also shows that this MDP cannot be reduced to a product MDP, since in product MDPs the best action for each agent does not depend on the policy of the other agents.

Assume two different policies $\pi_0^{(2)}, \pi_1^{(2)} : \mathcal{S} \to \Delta_{\mathcal{A}^{(2)}}$ for the second agent, where $\Delta_{\mathcal{A}^{(2)}}$ is the set of all distributions over $\mathcal{A}^{(2)}$. We define the first policy as $\pi_0^{(2)}(s_i) = \delta_0$ and the second as $\pi_1^{(2)}(s_i) = \delta_1$ for all $i \in [3]$ where δ_j for $j \in \{0, 1\}$ is the Dirac delta distribution (i.e. $\delta_j(a^{(2)}) = 1$ if $a^{(2)} = j$ and $\delta_j(a^{(2)}) = 0$ if $a^{(2)} \neq j$). When agent 2

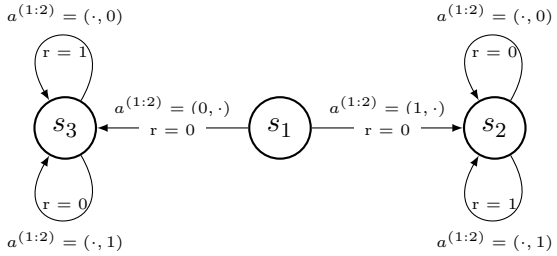

Figure 8.3: Illustration of an MDP for which Assumptions 1 and 4 hold.

follows $\pi_0^{(2)}$ the agents get a reward of 1 in s_3 and get a reward of 0 in s_2, regardless of the policy followed by the first agent. The effect of agent 2 following $\pi_1^{(2)}$ is exactly the opposite, the agents get a reward of 1 in s_2 and a reward of 0 in s_3. Consequently, the optimal action in state s_1 for agent 1 (since this determines the next state for both agents) depends on if $\pi_0^{(2)}$ or $\pi_1^{(2)}$ is used by the second agent. Importantly, this cannot be modeled by a product MDP, since in a product MDP the optimal action for each agent only depends on each agent's MDP, and does not depend on the policy of other agents. This example shows that there is an MDP, that is not a product MDP, for which Assumptions 1 and 4 still hold, strengthening the generality of our assumption.

In general we can construct an MDP for which Assumptions 1 and 4 are satisfied but is not a product MDP if at each state the transition probabilities and the reward only depend on the action of a single agent. The agent who's action is important at each state can change between states. One can think of this as having an agent that is in charge of what happens (transition and reward) at each state.

8.2.1 Verifying Assumption 1 Holds

We now show that the MDP in Fig. 8.3 satisfies Assumption 1. First we define the feature map $\phi : \mathcal{S} \times \mathcal{A}^{(1:2)} \to \mathbb{R}^4$.

$$\phi(s_1, (0,0)) = \phi(s_1, (0,1)) = [0,1,0,0]^\top$$
$$\phi(s_1, (1,0)) = \phi(s_1, (1,1)) = [1,0,0,0]^\top$$
$$\phi(s_2, (0,0)) = \phi(s_2, (1,0)) = [1,0,0,0]^\top$$
$$\phi(s_2, (0,1)) = \phi(s_2, (1,1)) = [1,0,1,0]^\top$$
$$\phi(s_3, (0,0)) = \phi(s_3, (1,0)) = [0,1,0,1]^\top$$
$$\phi(s_3, (0,1)) = \phi(s_3, (1,1)) = [0,1,0,0]^\top$$

We set $\gamma = 1/2$; however, the features can be changed to work for any $\gamma \in [0,1]$.

Now we show that all the deterministic policies satisfy Assumption 1 with the feature vectors above. Each policy can only have one of four action-value functions, which we show now. Notice that the action-vector taken by a policy in state s_1 can not change the action-value function, since the next state is deterministically s_2 or s_3 and there is no way to transition back to s_1. Also, the action selected by agent 1 does not affect the reward recieved in state s_2 or s_3. Thus, we are left to consider the policies that vary in the four combination of actions that are taken by agent 2 in state s_2 and s_3. We define these four polices as $\pi_{0,0}, \pi_{0,1}, \pi_{1,0}, \pi_{1,1}$, where $\pi_{x,y}$ defines any policy that takes action $x \in \{0,1\}$ in state s_2 and action $y \in \{0,1\}$ in s_3.

We claim that the following weight vectors satisfy Assumption 1, with $b = 3$.

$$w_{\pi_{0,0}} = [0,1,0,1]^\top \qquad w_{\pi_{0,1}} = [0,0,0,0]^\top$$
$$w_{\pi_{1,0}} = [1,1,1,1]^\top \qquad w_{\pi_{1,1}} = [1,0,1,0]^\top$$

We use $\cdot$ to indicate that the result holds for any action. For policy $\pi_{0,0}$ we have

$$
\begin{aligned}
w_{\pi_{0,0}}^{\top}\phi(s1,(0,\cdot)) &= [0,1,1,1][0,1,0,0]^{\top} = 1 && = q_{\pi_{0,0}}(s1,(0,\cdot)) \\
w_{\pi_{0,0}}^{\top}\phi(s1,(1,\cdot)) &= [0,1,1,1][1,0,0,0]^{\top} = 0 && = q_{\pi_{0,0}}(s1,(1,\cdot)) \\
w_{\pi_{0,0}}^{\top}\phi(s2,(\cdot,0)) &= [0,1,1,1][1,0,0,0]^{\top} = 0 && = q_{\pi_{0,0}}(s2,(\cdot,0)) \\
w_{\pi_{0,0}}^{\top}\phi(s2,(\cdot,1)) &= [0,1,1,1][1,0,1,0]^{\top} = 1 && = q_{\pi_{0,0}}(s2,(\cdot,1)) \\
w_{\pi_{0,0}}^{\top}\phi(s3,(\cdot,0)) &= [0,1,1,1][0,1,0,1]^{\top} = 2 && = q_{\pi_{0,0}}(s3,(\cdot,0)) \\
w_{\pi_{0,0}}^{\top}\phi(s3,(\cdot,1)) &= [0,1,1,1][0,1,0,0]^{\top} = 1 && = q_{\pi_{0,0}}(s3,(\cdot,1))
\end{aligned}
$$

For policy $\pi_{0,1}$ we have

$$
\begin{aligned}
w_{\pi_{0,1}}^{\top}\phi(s1,(0,\cdot)) &= [0,0,1,1][0,1,0,0]^{\top} = 0 && = q_{\pi_{0,1}}(s1,(0,\cdot)) \\
w_{\pi_{0,1}}^{\top}\phi(s1,(1,\cdot)) &= [0,0,1,1][1,0,0,0]^{\top} = 0 && = q_{\pi_{0,1}}(s1,(1,\cdot)) \\
w_{\pi_{0,1}}^{\top}\phi(s2,(\cdot,0)) &= [0,0,1,1][1,0,0,0]^{\top} = 0 && = q_{\pi_{0,1}}(s2,(\cdot,0)) \\
w_{\pi_{0,1}}^{\top}\phi(s2,(\cdot,1)) &= [0,0,1,1][1,0,1,0]^{\top} = 1 && = q_{\pi_{0,1}}(s2,(\cdot,1)) \\
w_{\pi_{0,1}}^{\top}\phi(s3,(\cdot,0)) &= [0,0,1,1][0,1,0,1]^{\top} = 1 && = q_{\pi_{0,1}}(s3,(\cdot,0)) \\
w_{\pi_{0,1}}^{\top}\phi(s3,(\cdot,1)) &= [0,0,1,1][0,1,0,0]^{\top} = 0 && = q_{\pi_{0,1}}(s3,(\cdot,1))
\end{aligned}
$$

For policy $\pi_{1,0}$ we have

$$
\begin{aligned}
w_{\pi_{1,0}}^{\top}\phi(s1,(0,\cdot)) &= [1,1,1,1][0,1,0,0]^{\top} = 1 && = q_{\pi_{1,0}}(s1,(0,\cdot)) \\
w_{\pi_{1,0}}^{\top}\phi(s1,(1,\cdot)) &= [1,1,1,1][1,0,0,0]^{\top} = 1 && = q_{\pi_{1,0}}(s1,(1,\cdot)) \\
w_{\pi_{1,0}}^{\top}\phi(s2,(\cdot,0)) &= [1,1,1,1][1,0,0,0]^{\top} = 1 && = q_{\pi_{1,0}}(s2,(\cdot,0)) \\
w_{\pi_{1,0}}^{\top}\phi(s2,(\cdot,1)) &= [1,1,1,1][1,0,1,0]^{\top} = 2 && = q_{\pi_{1,0}}(s2,(\cdot,1)) \\
w_{\pi_{1,0}}^{\top}\phi(s3,(\cdot,0)) &= [1,1,1,1][0,1,0,1]^{\top} = 2 && = q_{\pi_{1,0}}(s3,(\cdot,0)) \\
w_{\pi_{1,0}}^{\top}\phi(s3,(\cdot,1)) &= [1,1,1,1][0,1,0,0]^{\top} = 1 && = q_{\pi_{1,0}}(s3,(\cdot,1))
\end{aligned}
$$

For policy $\pi_{1,1}$ we have

$$
\begin{aligned}
w_{\pi_{1,1}}^{\top}\phi(s1,(0,\cdot)) &= [1,0,1,1][0,1,0,0]^{\top} = 0 && = q_{\pi_{1,1}}(s1,(0,\cdot)) \\
w_{\pi_{1,1}}^{\top}\phi(s1,(1,\cdot)) &= [1,0,1,1][1,0,0,0]^{\top} = 1 && = q_{\pi_{1,1}}(s1,(1,\cdot)) \\
w_{\pi_{1,1}}^{\top}\phi(s2,(\cdot,0)) &= [1,0,1,1][1,0,0,0]^{\top} = 1 && = q_{\pi_{1,1}}(s2,(\cdot,0)) \\
w_{\pi_{1,1}}^{\top}\phi(s2,(\cdot,1)) &= [1,0,1,1][1,0,1,0]^{\top} = 2 && = q_{\pi_{1,1}}(s2,(\cdot,1)) \\
w_{\pi_{1,1}}^{\top}\phi(s3,(\cdot,0)) &= [1,0,1,1][0,1,0,1]^{\top} = 1 && = q_{\pi_{1,1}}(s3,(\cdot,0)) \\
w_{\pi_{1,1}}^{\top}\phi(s3,(\cdot,1)) &= [1,0,1,1][0,1,0,0]^{\top} = 0 && = q_{\pi_{1,1}}(s3,(\cdot,1))
\end{aligned}
$$

It remains to show that the non-deterministic policies also satisfy Assumption 1. For a policy $\pi^{(p2,p3)}$ that takes action $(\cdot,1)$ at s_2 with probability $p_2 \in [0,1]$, and action $(\cdot,0)$ at s_3 with probability $p_3 \in [0,1]$, the weight vector that satisfies Assumption 1 with $b = 3$ is

$$w_{\pi} = [p_2, p_3, 1, 1]^{\top}$$

Verifying we see this is indeed true

$$
\begin{aligned}
w_{\pi^{(p2,p3)}}^{\top}\phi(s1,(0,\cdot)) &= [p_2,p_3,1,1][0,1,0,0]^{\top} = p_3 && = q_{\pi^{(p2,p3)}}(s1,(0,\cdot)) \\
w_{\pi^{(p2,p3)}}^{\top}\phi(s1,(1,\cdot)) &= [p_2,p_3,1,1][1,0,0,0]^{\top} = p_2 && = q_{\pi^{(p2,p3)}}(s1,(1,\cdot)) \\
w_{\pi^{(p2,p3)}}^{\top}\phi(s2,(\cdot,0)) &= [p_2,p_3,1,1][1,0,0,0]^{\top} = p_2 && = q_{\pi^{(p2,p3)}}(s2,(\cdot,0)) \\
w_{\pi^{(p2,p3)}}^{\top}\phi(s2,(\cdot,1)) &= [p_2,p_3,1,1][1,0,1,0]^{\top} = p_2+1 && = q_{\pi^{(p2,p3)}}(s2,(\cdot,1)) \\
w_{\pi^{(p2,p3)}}^{\top}\phi(s3,(\cdot,0)) &= [p_2,p_3,1,1][0,1,0,1]^{\top} = p_3+1 && = q_{\pi^{(p2,p3)}}(s3,(\cdot,0)) \\
w_{\pi^{(p2,p3)}}^{\top}\phi(s3,(\cdot,1)) &= [p_2,p_3,1,1][0,1,0,0]^{\top} = p_3 && = q_{\pi^{(p2,p3)}}(s3,(\cdot,1))
\end{aligned}
$$

Chapter 9

Conclusions and Future Work

In this thesis we considered the problem of planning with a local access simulator when the action space is large. We introduced several algorithms that achieve polynomial computation and query complexity guarantees, while still maintaining a reasonable suboptimality of the output policy under various assumptions. The main novelty is an efficient implementation of the UncertaintyCheck subroutine (required by Confident MC-LSPI and Confident MC-Politex) 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.

We provide Table 9.1, which summarizes the main results of this thesis. In bold it highlights the trade-offs between our algorithms (UncertaintyCheck-EGSS and UncertaintyCheck-DAV) and UncertaintyCheck-Naive by Yin et al. (2021). Recall that the objective of this thesis was to avoid computation complexity that depended on poly($|\mathcal{A}|$). Looking at the final column of the table, one can see that we have succeeded at this goal, since our algorithms require computation that does not depend on the action space or only depends on poly($\sum_{i=1}^{m} |A^{(i)}|$) (which is typically much smaller than $|\mathcal{A}| = \prod_{i=1}^{m} |A^{(i)}|$). To achieve this computational improvement only a poly(d, m) increase in the number of queries or suboptimality needs to be paid.

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

Table 9.1: Query complexity, computation complexity, and suboptimality bounds of algorithms discussed in this thesis, under Assumptions 1 and 2. The algorithms LSPI and POLITEX refer to CONFIDENT MC-LSPI and CONFIDENT MC-POLITEX respectively. NAIVE, EGSS, and DAV refer to UNCERTAINTYCHECK-NAIVE, UNCERTAINTYCHECK-EGSS, UNCERTAINTYCHECK-DAV respectively. The bold terms indicate the trade-offs between our algorithms (UNCERTAINTYCHECK-EGSS and UNCERTAINTYCHECK-DAV) and UNCERTAINTYCHECK-NAIVE from Yin et al. (2021). For $\epsilon = 0$, the suboptimality gap is $\kappa > 0$, while for $\epsilon > 0$, the suboptimality gap is given in the "Subopt $\epsilon > 0$" column. All algorithms also require $\text{poly}(\frac{1}{1-\gamma}, \frac{1}{\kappa}, \log(\frac{1}{\delta}))$ computation for $\epsilon = 0$ and $\text{poly}(\frac{1}{1-\gamma}, \frac{1}{\epsilon}, \log(\frac{1}{\delta}), \log(1+b))$ computation for $\epsilon > 0$. UNCERTAINTYCHECK-EGSS requires access to a greedy oracle (Assumption 3). UNCERTAINTYCHECK-DAV assumes the action set and feature map is additive (Assumption 4).

An interesting direction for future work is to extend our results to the CONFIDENT LSVI algorithm (Hao et al., 2022), which is an algorithm based on least-squares value iteration (LSVI) that only assumes a local access simulator and also uses UNCERTAINTYCHECK-NAIVE to construct a core set of features. Further, as discussed at the end of Chapter 1, the recent CAPI-QPI-PLAN algorithm by Weisz et al. (2022) achieves better results than CONFIDENT MC-LSPI or CONFIDENT MC-POLITEX with local access to a simulator; however, its computation also depends on $|\mathcal{A}|$ and could potentially be extended to large action sets using a variation of our UNCERTAINTYCHECK algorithms. In this thesis we assumed a deterministic initial state, as the focus was on addressing large action spaces; however, this can likely be extended to a random initial state, as was done by Yin et al. (2021).

Bibliography

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.

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.

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.

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.

Gellért Weisz, András György, Tadashi Kozuno, and Csaba Szepesvári. Confident approximate policy iteration for efficient local planning in q pi-realizable mdps. *Advances in Neural Information Processing Systems*, 35:25547–25559, 2022.

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 powergrid 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.

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. Multi-agent 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 multi-agent 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*, 2021.

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.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Jack Kiefer and Jacob Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960.

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.

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

Appendix A: Parameter Settings

A.1 Parameter Settings for Confident MC-LSPI + UncertaintyCheck-Naive

If $\epsilon = 0$

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

If $\epsilon > 0$, then set $\kappa = \frac{32\epsilon\sqrt{d}}{(1-\gamma)^2}(1 + \log(b^2\epsilon^{-2}d^{-1}))^{1/2}$ in the above displays.

A.2 Parameter Settings for Confident MC-LSPI + UncertaintyCheck-EGSS

If $\epsilon = 0$

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

If $\epsilon > 0$, then set $\kappa = \frac{32\epsilon d}{(1-\gamma)^2}(1 + \log(b^2\epsilon^{-2}d^{-1}))^{1/2}$ in the above displays.

A.3 Parameter Settings for Confident MC-LSPI + UncertaintyCheck-DAV

If $\epsilon = 0$

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

If $\epsilon > 0$, then set $\kappa = \frac{32\epsilon\sqrt{d}m}{(1-\gamma)^2}(1 + \log(b^2\epsilon^{-2}d^{-1}))^{1/2}$ in the above displays.

A.4 Parameter Settings for Confident MC-Politex + UncertaintyCheck-Naive

If $\epsilon = 0$

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

If $\epsilon > 0$, then set $\kappa = \frac{16\epsilon\sqrt{d}\zeta}{(1-\gamma)}(1 + \log(b^2\epsilon^{-2}d^{-1}))^{1/2}$ in the above displays.

A.5 Parameter Settings for Confident MC-Politex + UncertaintyCheck-EGSS Parameters

If $\epsilon = 0$

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

If $\epsilon > 0$, then set $\kappa = \frac{16\epsilon d}{(1-\gamma)}(1 + \log(b^2\epsilon^{-2}d^{-1}))^{1/2}$ in the above displays.

A.6 Parameter Settings for Confident MC-Politex + UncertaintyCheck-DAV Parameters

If $\epsilon = 0$

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

If $\epsilon > 0$, then set $\kappa = \frac{16\epsilon\sqrt{dm}}{(1-\gamma)}(1 + \log(b^2\epsilon^{-2}d^{-1}))^{1/2}$ in the above displays.