Safe Multi-Agent Reinforcement Learning in Polynomial Time

by

Jeremy McMahan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science)

at the
University of Wisconsin-Madison
2025

Date of Final Oral Exam: 08/13/2025
The dissertation is approved by the following members of the Final Oral Committee:

Jerry Zhu, Professor, Computer Sciences
Yudong Chen, Associate Professor, Computer Sciences
Josiah Hanna, Assistant Professor, Computer Sciences
Sandeep Silwal, Assistant Professor, Computer Sciences
Qiaomin Xie, Assistant Professor, Industrial and Systems Engineering

Contents

List of Figures

Chapter 1

Introduction

Background. The Markov Decision Process (MDP), introduced by Richard Bellman as part of his foundational work on dynamic programming, has become one of the most powerful tools for modeling decision-making under uncertainty. Just as dynamic programming provides a general algorithmic strategy for solving sequential problems, MDPs offer a remarkably expressive formalism for capturing a wide array of optimization tasks. Classical problems such as the knapsack problem and the clique problem, among others, can be encoded within the MDP framework. In fact, with sufficient modeling flexibility, many discrete optimization problems can be cast as MDPs and approximately solved using standard techniques. Reinforcement learning (RL) further extends the power of this framework by enabling the solution of such problems through repeated interaction with an environment, even in the absence of an explicit model. The extension to Markov games (MG), multi-agent generalizations of MDPs, broadens the scope even further to include competitive, cooperative, and general multi-agent systems. Unsurprisingly, a framework of this generality and power has become central to a wide range of real-world applications, including robotics, finance, recommendation systems, online advertising, and autonomous control.

Safe RL. However, when deployed in real-world environments, the standard Markov game model may fail to capture the full spectrum of risks necessary to ensure agent safety. For instance, an opponent may strategically conceal its true objectives to manipulate an agent into dangerous or suboptimal behavior. In more severe cases, an external attacker — or even benign but unpredictable sensory noise — can disrupt the agent-environment interaction in subtle but critical ways. Consider, for example, a nefarious actor who strategically and iteratively obscures road signs to mislead an autonomous vehicle to a dangerous location. Even in the absence of adversaries, the environment itself may present latent hazards that are not easily encoded within a standard MG formulation. For example, a requirement to ensure a low probability of tire failure over repeated delivery routes introduces complex probabilistic constraints that fall outside the expressiveness of the basic framework since reward shaping cannot induce precise probabilities of success.

The Safety Landscape. These examples illustrate that we must expand beyond the standard model if agent safety is to be guaranteed. Generally, we see that safety concerns manifest in two key ways, each corresponding to a fundamental entity of the MG: threats from other *agents* and threats from the *environment* itself. The first category — safety against adversarial agents or malicious attackers — is the central focus of *adversarial reinforcement learning* (adversarial RL). Here, the goal is to compute policies that are robust to worst-case opponents or external perturbations. The second category — safety against environmental hazards or operational requirements — is typically modeled through constraints and studied within *constrained reinforcement learning* (constrained RL). In this setting, the goal is to compute policies that satisfy constraints while maximizing the expected return. Together, these two lines of research address both essential considerations necessary

for deploying autonomous agents in high-stakes, uncertain environments.

Prior Work. Despite the importance of these fields, many fundamental open problems have remained unresolved. On the adversarial RL side, prior work had established how to compute optimal perceived-state attacks and had developed no-regret strategies for reward poisoning. However, the general problem of designing optimal attacks and, importantly, computing robust defense policies, remained largely open. In addition, more realistic misinformation attacks, in which opponent agents strategically misreport their rewards, had not been formally studied. On the constrained RL side, existing results were limited to the computation of optimal stochastic policies under expectation constraints. As a result, the foundational question of efficiently computing safe deterministic policies or policies satisfying more general classes of constraints remained unanswered for nearly 25 years.

Our Contributions. In this work, we aim to make progress on, if not entirely resolve, each of the open questions outlined above. The first two chapters focus on problems in adversarial RL, specifically the derivation of polynomial-time algorithms for both attack and defense settings. The final three chapters address questions in constrained RL, culminating in polynomial-time algorithms for computing near-optimal deterministic policies that satisfy a multitude of constraint criteria. Here, we focus on the single agent case in the first two chapters and then extend to the full multi-agent setting in the last chapter. A breakdown of the main contributions from each chapter follows.

1. In Chapter 2, we study classic manipulation attacks in multi-agent reinforcement learning (MARL), where an opponent agent or external attacker can directly alter components of the agent-environment interaction, such as observations

and rewards. This chapter culminates in polynomial-time algorithms for the attack problem under arbitrary attack surfaces, and for the defense problem under any combination of non-perceived-state attacks. This work appeared in AAAI 2024.

2. In Chapter 3, we study misinformation attacks in MARL. In this setting, an opponent player in a two-player game can strategically share a falsified reward function with the victim player. The opponent's objective is to constrain the victim's rational responses to maximize their own payoff. This chapter culminates in a polynomial-time algorithm for computing the optimal fake reward function, when restricted to the class of dominant-column reward matrices. This work appeared in RLC 2024.

3. In Chapter 4, we introduce a new type of constraint, called an *anytime constraint*, designed to ensure strict safety. An anytime constraint requires that the cumulative cost incurred by a policy remains within budget at all time steps. This chapter culminates in a polynomial-time $(0, \epsilon)$-bicriteria approximation algorithm for computing anytime-compliant policies. This work appeared in AISTATS 2024.

4. In Chapter 5, we study the computation of deterministic policies for general constrained MDPs with a single constraint. Here, we consider a class of constraints characterized by generalized policy evaluation equations, which includes expectation, almost-sure, and anytime constraints. This chapter culminates in a fully polynomial-time approximation scheme (FPTAS) for computing deterministic, constrained policies. This work appeared in NeurIPS 2024.

5. In Chapter 6, we extend the constrained policy computation framework to

general constrained MGs with an arbitrary number of constraints. The constraint model considered includes all known variants studied in the literature, including chance constraints. This chapter culminates in a polynomial-time $(0, \epsilon)$-bicriteria approximation algorithm for computing constrained policies. This work appeared in ICML 2025.

Collectively, these results advance the theoretical foundations of safe reinforcement learning by providing the first polynomial-time algorithms for a range of adversarial and constrained decision-making settings.

Chapter 2

Manipulation Attacks

Acknowledgments. This chapter was a joint work with Young Wu, Jerry Zhu, and Qiaomin Xie that appeared in AAAI 2024.

Abstract. To ensure the usefulness of Reinforcement Learning (RL) in real systems, it is crucial to ensure they are robust to noise and adversarial attacks. In adversarial RL, an external attacker has the power to manipulate the victim agent's interaction with the environment. We study the full class of online manipulation attacks, which include (i) state attacks, (ii) observation attacks (which are a generalization of perceived-state attacks), (iii) action attacks, and (iv) reward attacks. We show the attacker's problem of designing a stealthy attack that maximizes its own expected reward, which often corresponds to minimizing the victim's value, is captured by a Markov Decision Process (MDP) that we call a meta-MDP since it is not the true environment but a higher level environment induced by the attacked interaction. We show that the attacker can derive optimal attacks by planning in polynomial time or learning with polynomial sample complexity using standard RL techniques. We argue that the optimal defense policy for the victim can be computed as the solution to a stochastic Stackelberg game, which can be further simplified into a

partially-observable turn-based stochastic game (POTBSG). Neither the attacker nor the victim would benefit from deviating from their respective optimal policies, thus such solutions are truly robust. Although the defense problem is NP-hard, we show that optimal Markovian defenses can be computed (learned) in polynomial time (sample complexity) in many scenarios.

2.1 Introduction

Reinforcement Learning (RL) has become a staple with a plethora of applications including the breakthrough ChatGPT [85]. With the growth of RL applications, it is critical to understand the security threats posed to RL and how to defend against them. In many applications, noisy measurements can cause the agent-environment interaction to evolve entirely differently than what one would expect in theory. Even worse, malicious attackers can strategically modify the agent-environment interaction to induce catastrophic outcomes for the agent. If RL methods are to be used in diverse and critical settings, it is essential to ensure these RL algorithms are robust to potential attacks.

In adversarial RL, a victim agent interacts with an environment while being disrupted by an attacker. The attacker has the power to manipulate each aspect of the victim-environment interaction. In particular, the attacker can change: (i) the environment's state (*state attacks*), (ii) the victim's observation (*observation attacks*), (iii) the action taken by the victim (*action attacks*), and (iv) the reward received by the victim (*reward attacks*). When the environment is fully-observable, observation attacks translate to well-studied *perceived-state attacks*. We refer to all of these attack surfaces by *online manipulation attacks*. The attacker may use a subset or all of these attack surfaces to optimize its own expected reward from

the attack, which often corresponds to minimizing the victim's value. However, the attacker cannot perform arbitrary manipulations without raising suspicion. Hence, the attacker must restrict its manipulations to a predefined set of stealthy attacks. On the other hand, the attacker-aware victim seeks to choose a *defense* policy whose value is provably robust even under the worst possible stealthy attack.

From the attacker's perspective, it faces an optimal control problem: it needs to strategically choose stealthy attacks to optimize its value. Unlike typical control problems, the attacker must deal with the uncertainty of the victim's actions in addition to that of the stochastic environment. Thus, the attacker's problem involves a multi-agent feature. For any fixed victim policy π, we can view the attacker's problem as computing its best response attack to the victim's chosen π. From the victim's perspective, we argue it faces a Stochastic Stackelberg game: it needs to choose a policy that achieves maximum value in the environment under the attacker's best-response attack. A defense policy designed following this principle ensures neither the victim nor the attacker would benefit from deviating from their chosen policies, and so an equilibrium would be achieved. This implies that regardless of the attack, the defense policy always achieves at least the game's optimal value. However, computing optimal Stackelberg strategies for stochastic games is NP-hard. Thus, both the attacker and the victim are faced with challenging optimization problems.

Although the attack and defense problems are of great importance, complete solutions have yet to be discovered. For the attack problem, most works focus on the empirical aspects, lacking theoretical guarantees. Provably optimal attacks have only been devised for the special case of test-time, perceived-state attacks [99, 129, 104]. The situation is even worse for the defense problem, which is arguably more important. Nearly all proposed defenses are designed to be effective against a specific, known attack. This results in a cat-and-mouse game: the attacker can just design a new

attack for the given defense policy and so the victim would always be at risk. In addition, the two approaches with provable guarantees are restricted to the planning, reward-poisoning setting [8], and the test-time, perceived-state attack setting [130]. Furthermore, neither defense can be computed efficiently and it is unrealistic to assume the victim knows the attacker's exact algorithm.

Our Contributions. Despite the challenges of the attack and defense problems, we develop frameworks for computing optimal attacks and defenses for any combination of attack surfaces, which are provably efficient in many cases. From the attacker's side, we show that for any fixed victim policy, the optimal attack can be computed as the solution to another Markov Decision Process (MDP). We call this environment a *meta-MDP* since it is not the true environment, but is a higher-level environment induced by the victim-attacker-environment interaction. Importantly, the attacker can simulate an interaction with the meta-MDP by interacting with the victim and the true environment. Hence, the attacker can attack optimally by solving the meta-MDP using any standard MDP planning or RL algorithms. In addition, we show that the size of the meta-MDP is polynomial in the size of the original environment and the size of the victim's policy. Thus, optimal attacks can always be computed or learned efficiently. Our framework also extends to linear MDPs. Hence, we provide the first provably optimal attacks for beyond perceived-state attacks and the first provably optimal attacks for the linear setting, all of which can be computed in polynomial time. We note our framework also solves the certifying robustness problem posed in [118].

On the victim's side, we argue that the defense problem is most naturally modeled by a stochastic Stackelberg game [113], which can be captured by a much simpler partially-observable turn-based stochastic game (POTBSG) [49]. Thus, the victim

can compute its optimal robust defense by finding a weak Stackelberg equilibrium (WSE) for the meta-POTBSG. Again, the victim can simulate the meta-POTBSG by interacting with the attacker and the true environment. When the attacker is adversarial, the victim can defend optimally by solving the meta-POTBSG using any standard zero-sum POTBSG planning or distributed learning algorithms. Unlike the attack problem, we show that the victim's defense problem is NP-hard in general even to find approximate solutions when observation attacks are permitted. However, we show that optimal Markovian defenses can be computed efficiently when excluding observation attacks by exploiting the sequential nature of the attacks. This gives a broad class of games for which WSE is computable. Overall, we present the first-ever provable defense algorithms for both the planning and learning settings and show our defenses can be computed efficiently for a broad class of instances.

2.1.1 Related Work

Many prior works have studied adversarial RL under various models and objectives. Amongst the first works, Behzadan and Munir [9], Huang et al. [58], Kos and Song [66] study perceived-state attacks through the lens of adversarial examples for deep neural nets [42]. Kos and Song [66] also considers adversarial examples, but with the goal of minimizing the number of attacks needed to achieve large damage. These works focused on achieving large damage at the current time. Later Lin et al. [71], Sun et al. [103] developed more advanced heuristics that incorporate future value into their attacks to achieve long-term damage. Meanwhile, Tretschk et al. [108] trained an adversarial deep net to compute perturbations that allows other objectives for the attacker.

Afterward, many works began considering the objective of maximizing the damage to the victim rather than minimizing the number of attacks. Russo and Proutiere

[99], Zhang et al. [129] developed optimal algorithms for computing perceived-state attacks. Both works formulated the attack problem as a different MDP as we do here. Sun et al. [104] formulated an actor-director model for the attack problem that is easier to solve for some MDPs and retains guarantees of optimality. The idea of adversarial training was then used in conjunction with the attack formulation from [129] to obtain experimentally robust victim policies [130].

Action and reward attacks have been considered heavily in the training-time setting. For example, Tessler et al. [106], Lee et al. [67] considered action attacks. Reward poisoning attacks are the focus of the work by Zhang et al. [131], Rangi et al. [95]. In fact, a combination action and reward attack are devised by Rangi et al. [95]. Most of these works consider the policy teaching setting, where the attacker's goal is for the victim to follow a fixed policy $\pi^\dagger$. Some algorithms achieve sublinear regret for the attacker when the victim policy is no regret [72]; though, none compute truly optimal attacks.

2.2 Attack Surfaces

POMDPs. We denote a infinite-horizon discounted environment POMDP by $M = (\mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \gamma, \mu)$ where (i) $\mathcal{S}$ is the state set, (ii) $\mathcal{O}$ is the observation set, (iii) $\mathcal{A}$ is the action set, (iv) $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, (v) $R : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward distribution, (vi) γ is the discount factor, and (vii) $\mu \in \Delta(S)$ is the initial state distribution. We let $\mathcal{O}(s)$ denote the distribution of observations at state s. We also let $\mathcal{R}$ denote the set of all supported rewards. The total expected reward the victim receives from following policy π in environment M is its *value*, i.e., the expected cumulative discounted rewards $V_M^\pi := \mathbb{E}_M^\pi \left[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)\right]$.

Suppose the victim interacts with a Markovian environment, M, using a fixed

stationary, Markovian policy $\pi : \mathcal{O} \to \Delta(A)$. At any time t, let s_t denote M's current state and o_t denote the generated observation. In the standard setting, the victim chooses an action $a_t \sim \pi(o_t)$ and then receives a reward $r_t \sim R(s_t, a_t)$. Afterwards, M transitions to its next state $s_{t+1} \sim P(s_t, a_t)$. We see there are several points during time t at which information is exchanged between the victim and M. We further break down the interaction at time t based on these points of information exchange, which we call *subtimes*:

1. At the first subtime, t_1, M receives its state $s_t \sim P(s_{t-1}, a_{t-1})$.
2. At the second subtime, t_2, the victim receives its observation $o_t \sim \mathcal{O}(s_t)$.
3. At the third subtime, t_3, M receives the victim's action $a_t \sim \pi(o_t)$.
4. At the fourth subtime, t_4, the victim receives its reward $r_t \sim R(s_t, a_t)$.

Online Attacks. In the adversarial setting, a third-party called the *attacker* interferes with the victim-M interaction. Here, the attacker may intercept and then corrupt the information being exchanged between the victim and environment M. The attacker has access to four attack surfaces:

1. (*State Attack*) A state attack changes the state of M from s_t to $s_t^\dagger$. The attack influences the observation $o_t \sim \mathcal{O}(s_t^\dagger)$. If M receives action a_t, the attack also influences the reward $r_t \sim R(s_t^\dagger, a_t)$ and the next state $s_{t+1} \sim P(s_t^\dagger, a_t)$.
2. (*Observation Attack*) An observation attack causes the victim to receive observation $o_t^\dagger$ instead of $o_t \sim \mathcal{O}(s_t)$. The attack influences the victim's action $a_t \sim \pi(o_t^\dagger)$.
3. (*Action Attack*) An action attack causes M to receive action $a_t^\dagger$ instead of a_t. The attack influences the reward $r_t \sim R(s_t, a_t^\dagger)$ and next state $s_{t+1} \sim P(s_t, a_t^\dagger)$.

4. (*Reward Attack*) A reward attack causes the victim to receive reward $r_t^\dagger$ instead of reward $r_t \sim R(s_t, a_t)$.

We call each of these attack surfaces as *online manipulation attacks*. These attack surfaces in conjunction give the attacker the power to corrupt every element of the triple (s, a, r) that define the interaction between the victim and M.

If M is fully observable, observation attacks correspond to *perceived-state* attacks, which change what the victim thinks is M's state. Notice unlike the other surfaces, state attacks could be performed at two different subtimes. Namely, the attacker can change the state before M transitions at t_1 or before M receives the victim's action at t_3. For simplicity, we assume state attacks only happen at t_1, but our results apply equally well to both versions.

Adversarial Interaction. Overall, the victim-attacker-M interaction at time t now evolves as follows:

1. At subtime t_1, M is in state s_t.
   (a) Attacker: changes s_t to $s_t^\dagger$.
   (b) M: enters state $s_t^\dagger$ and generates observation $o_t \sim \mathcal{O}(s_t^\dagger)$.
2. At subtime t_2, M is in state $s_t^\dagger$ and has generated observation o_t.
   (a) Attacker: changes o_t to $o_t^\dagger$.
   (b) Victim: chooses action $a_t \sim \pi(o_t^\dagger)$.
3. At subtime t_3, M is in state $s_t^\dagger$ and the victim chose action a_t.
   (a) Attacker: changes a_t to $a_t^\dagger$.
   (b) M: generates reward $r_t \sim R(s_t^\dagger, a_t^\dagger)$ and generates state $s_{t+1} \sim P(s_t^\dagger, a_t^\dagger)$.

4. At subtime t_4, M has generated reward r_t.

   (a) Attacker: changes r_t to $r_t^\dagger$.

   (b) Victim: receives reward $r_t^\dagger$.

This process then repeats starting from s_{t+1}.

Attacker Constraints. In general, the attacker may not arbitrarily manipulate the interaction. For example, some attacks may be physically impossible or risk detection. As such, we assume the attacker has a set $\mathcal{B}$ that defines the feasible manipulations it can perform. For example, the attacker might require a manipulated observation to be visually similar to the true observation. Thus, the set of feasible observation attacks should depend on the true observation. Applying the same logic to each attack surface, we see the feasible attack sets should take the form: $\mathcal{B}(s) \subseteq \mathcal{S}$, $\mathcal{B}(o) \subseteq \mathcal{O}$, $\mathcal{B}(a) \subseteq \mathcal{A}$, and $\mathcal{B}(r) \subseteq \mathcal{R}$. However, in some cases, the feasibility of an attack would depend on the interaction before the attack, not just the current element being manipulated. To be fully general, we allow the feasibility sets to take the form: at subtime t_1, $\mathcal{B}(s) \subseteq \mathcal{S}$; at subtime t_2, $\mathcal{B}(s, o) \subseteq \mathcal{O}$; at subtime t_3, $\mathcal{B}(s, o, a) \subseteq \mathcal{A}$; and, at subtime t_4, $\mathcal{B}(s, o, a, r) \subseteq \mathcal{R}$.

2.3 Optimal Attacks

Attacker's Goal. We saw how an attacker can disrupt an interaction but have yet discussed why it would do this. Suppose the attacker has a reward function $g(s, a, r)$ that depends on the victim's received reward, possibly in addition to M's state and the victim's action. The attacker's goal is then to construct an attack that maximizes its own expected reward. Commonly, an attacker just wants to minimize the victim's expected reward under attack, or equivalently maximize the

damage to the victim's expected reward. In this case, the attacker's reward function is $g(s, a, r) = -r$. Alternatively, the attacker may want the victim to behave in a specified way. This goal is equivalent to the attacker wanting the victim to choose actions that match a fixed target policy $\pi^\dagger$ as often as possible. In this case, the attacker's reward function is $g(s, a, r) = \mathbf{1}\left\{a = \pi^\dagger(s)\right\}$.

Definition 1 (Attack Problem)**.** For any π, the attacker's seeks a policy $\nu^* \in N$ that maximizes its expected reward from the victim-attacker-M interaction:

$$\nu^* \in \arg\max_{\nu \in N} \mathbb{E}_M^{\pi,\nu}\left[\sum_{t=0}^{\infty} \gamma^t g(s_t, a_t, r_t)\right]. \tag{2.1}$$

We show that the attacker's problem is captured by a MDP. The key insight is that by defining the attacker's state set to capture the results of previous attacks from t_1 up to the current subtime, then each attack becomes Markovian with respect to the expanded state set. This is not a significant burden on the attacker since it would need to keep track of this information anyway to compute the feasible attack sets. Thus, the attacker just needs to keep track of the information within a time step to compute optimal attacks.

Definition 2 (Meta-MDP)**.** For any victim policy π, the attacker's meta-MDP is $\bar{M} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{P}, \bar{r}, \overline{\gamma}, \bar{\mu})$ where,

- $\bar{\mathcal{S}} = \mathcal{S} \cup (\mathcal{S} \times \mathcal{O}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A} \times \mathcal{R})$.
- $\bar{\mathcal{A}}(s) = \mathcal{B}(s)$, $\bar{\mathcal{A}}(s, o) = \mathcal{B}(s, o)$, $\bar{\mathcal{A}}(s, o, a) = \mathcal{B}(s, o, a)$, and $\bar{\mathcal{A}}(s, o, a, r) = \mathcal{B}(s, o, a, r)$.
- The transitions vary per subtime. Let $\bar{s} \in \bar{\mathcal{S}}$, $\bar{a} \in \bar{\mathcal{A}}(\bar{s})$, and $\bar{s}' \in \bar{\mathcal{S}}$.

    1. If $\bar{s} = s$, then $\bar{a} = s^\dagger$ and $\bar{s}' = (s^\dagger, o)$: $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) = \mathcal{O}(o \mid s^\dagger)$.

2. If $\bar{s} = (s, o)$, then $\bar{a} = o^\dagger$ and $\bar{s}' = (s, o^\dagger, a)$: $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) = \pi(a \mid o^\dagger)$.
3. If $\bar{s} = (s, o, a)$, then $\bar{a} = a^\dagger$ and $\bar{s}' = (s, o, a^\dagger, r)$: $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) = R(r \mid s, a^\dagger)$.
4. If $\bar{s} = (s, o, a, r)$, then $\bar{a} = r^\dagger$ and $\bar{s}' = s'$: $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) = P(s' \mid s, a)$.

All other transitions have probability 0.

- Let $\bar{s} \in \bar{\mathcal{S}}$, and $\bar{a} \in \bar{\mathcal{A}}(\bar{s})$. If $\bar{s} = (s, o, a, r)$ and $\bar{a} = r^\dagger$, then $\bar{r}(\bar{s}, \bar{a}) = g(s, a, r^\dagger)$. For all other meta-states, $\bar{r}(\bar{s}, \bar{a}) = 0$.
- $\overline{\gamma} = \gamma^{1/4}$.
- $\bar{\mu}(s) = \mu(s)$ for $s \in \mathcal{S}$ and $\bar{\mu}(\bar{s}) = 0$ otherwise.

Reward Subtlety. Note that the attacker only receives a reward at every fourth subtime. This means the discount factor has to be "slowed down" so that the factor at every fourth time step matches that of each single time step of M. Specifically, choosing $\overline{\gamma} = \gamma^{1/4}$ ensures that $\overline{\gamma}^{4t} = \gamma^t$.

Proposition 1. *The maximum expected reward the attacker can achieve from any attack on π is $V^*_{\bar{M}}$, the maximum expected total discounted reward for the meta-MDP $\bar{M}$. Furthermore, any optimal deterministic, stationary policy ν^* for $\bar{M}$ is an optimal attack policy.*

Online Interaction. Suppose the attacker has computed some attack policy ν from $\bar{M}$. In order to use ν to interact with the victim and M, the attacker must know the meta-state at any given subtime. As long as the attacker can observe the interaction between the victim policy π and M, it can effectively simulate the interaction with the meta-MDP $\bar{M}$ online using a constant amount of memory. At time t, the attacker only needs to store s_t, o_t, a_t, and r_t when they are revealed to

Algorithm 1 Attacker Interaction Protocol

Input: (π, ν)

for $t = 1 \ldots$ **do**
  Attacker sees s_t, and computes a state attack $s_t^\dagger = \nu(s_t)$
  Attacker sees $o_t \sim \mathcal{O}(s_t^\dagger)$, and computes $o_t^\dagger = \nu(s_t, o_t)$
  Attacker sees $a_t \sim \pi(o_t^\dagger)$, and computes $a_t^\dagger = \nu(s_t, o_t, a_t)$
  Attacker sees $r_t \sim \mathcal{R}(s_t^\dagger, a_t^\dagger)$, and computes $r_t^\dagger = \nu(s_t, o_t, a_t, r_t)$
  Attacker receives reward $g(s_t^\dagger, a_t^\dagger, r_t^\dagger)$, and forgets (s_t, o_t, a_t, r_t)

the attacker. With this information, the attacker knows the meta-state for each subtime and so can apply ν to determine its next attack. Upon reach the next time $t+1$, the attacker can forget s_t, o_t, a_t, and r_t and start from s_{t+1}. See Algorithm 1.

Solving $\bar{M}$. If the attacker has full knowledge of M and the victim's policy π, then the attacker has all the knowledge needed to construct the meta-MDP $\bar{M}$. Once $\bar{M}$ is constructed, the attacker can use any planning algorithm, such as policy iteration, to compute the optimal attack. Alternatively, if the attacker does not know M and π, it can still simulate interacting with $\bar{M}$ online as described before to perform learning. In particular, the attacker can replace the call to ν in Algorithm 1 with any off-the-shelf learning algorithm. For the episodic setting, we view the attacker as attacking a new victim following the same policy π in each episode.

Observe that $|\bar{\mathcal{S}}| \leq |\mathcal{S}||\mathcal{O}||\mathcal{A}||\mathcal{R}|$, $|\bar{\mathcal{A}}| \leq |\mathcal{S}| + |\mathcal{O}| + |\mathcal{A}| + |\mathcal{R}|$, and $\overline{\gamma} = \gamma^{1/4}$. Thus, whenever M's rewards are finitely supported, $|\bar{M}| = poly(|M|)$, where $|M|$ is the total size of M's description. As such, any polytime planning algorithm or polynomial sample-complexity learning algorithm applied to $\bar{M}$ yields an algorithm for computing optimal attacks that has polynomial complexity.

Proposition 2. *When M's rewards have finite support or no reward attacks are allowed, $|\bar{M}| = poly(|M|)$. Thus, an optimal attack policy can be computed in*

polynomial time by planning in $\bar{M}$, and learning an optimal attack policy can be performed with polynomial sample complexity by learning in $\bar{M}$.

Remark 1 (Restricted Surfaces). By restricting $\bar{\mathcal{A}}$ to singleton sets (e.g. set $\bar{\mathcal{A}}(s, o, a) = \{a\}$ to disallow action attacks), $\bar{M}$ recovers optimal attacks for each individual surface as well as attacks for any subset of available attack surfaces. This captures all standard test-time attacks, generalizing the perceived-state attack MDP of [129]. We also note if the attacker does not perform reward attacks, $\bar{M}$ can be modified to avoid $\mathcal{R}$ and so M having finite supported rewards is unnecessary in the complexity results.

One might ask whether the perceived-state attack MDP defined in [129] would work in the linear setting. We point out that the transition takes the following form,

$$\begin{aligned}\tilde{P}(s' \mid s, s^\dagger) &= \mathbb{E}_{a\sim\pi(s^\dagger)} P(s' \mid s, a) \\ &= \int_a P(s' \mid s, a)\pi(a \mid s^\dagger)da.\end{aligned}$$

As π and P are multiplied together, $\tilde{P}$ would be a quadratic transition. On the other hand, our particular choice of subtimes induces linear structure in $\bar{M}$. Specifically, each transition of $\bar{P}$ is defined by a single distribution involving π or M. If both π and M have a linear structure, then so will $\bar{M}$. Then, $\bar{M}$ can be solved by standard linear RL algorithms. Thus, so long as π is linear, the attacker can compute optimal attacks on linear environments.

Theorem 1. *If M is linear and π is linear, then $\bar{M}$ is linear. Furthermore, the dimension of $\bar{M}$, $d(\bar{M})$, is at most* $\max\{d(\pi), d(M)\}+1$. *Thus, if π is linear, optimal attacks on linear environments can be computed or learned efficiently*

Remark 2 (Beyond Markovian Policies). Our construction can be easily modified to handle non-Markovian victim policies. If the victim uses some finite amount of past

history $\tilde{\mathcal{H}}$, we simply modify the meta-state space to remember the same amount of past history and adjust the construction appropriately. The size of $\bar{M}$ is now a polynomial in both $|M|$ and the size of the policy when described explicitly as a mapping from histories to action distributions. We defer the details to the Appendix.

2.4 Optimal Defense

Now that we have seen how the attacker can best attack, it begs the question of how the victim should defend against attacks. Intuitively, the victim should choose a defense policy that is robust to attack. However, it does not suffice to just be robust against a particular attack. In fact, the attacker could lie about its attack algorithm to bait the victim into choosing a policy that actually benefits the attacker. Even if some attacker does use that particular attack algorithm, other attackers may employ different methods that lead the victim to poor value. As new attacks are formulated, the victim would have to constantly create more complex policies designed with all known attacks in mind. This would become a never-ending cat-and-mouse game during which the victim's policy will often be at risk of new attacks. Thus, for a policy to be satisfactorily robust, we require it to be robust against the worst possible attack. This way, no matter what future strategies an attacker may use, the victim is already prepared.

We can formalize this intuition using the Stackelberg approach for Security Applications [65]. For any π and ν, let $V_1^{\pi,\nu}$ and $V_2^{\pi,\nu}$ denote the victim's and attacker's expected reward respectively under the victim-attacker-M interaction induced by π and ν. Note, both of these quantities can be computed efficiently using the previous section's techniques. Let V_1 and V_2 denote infinite matrices whose (π, ν) entry corresponds to $V_1^{\pi,\nu}$ and $V_2^{\pi,\nu}$ respectfully. We define an infinite bimatrix

game G whose payoff matrices are (V_1, V_2). For any fixed victim π, it is clear that a rational attacker would play some best-response policy, $\nu \in BR(\pi) := \max_{\nu \in N} V_2^{\pi,\nu}$. Thus, an optimal defense policy is exactly an optimal Stackelberg strategy for player 1 in G [25].

Definition 3 (Defense Problem)**.** The victim seeks a policy π^* that maximizes its expected reward from the victim-attacker-M interaction under the worst-case attack:

$$\pi^* \in \max_{\pi \in \Pi} \min_{\nu \in BR(\pi)} V_1^{\pi,\nu}. \tag{2.2}$$

Observe that this solution is truly robust: by definition, the attacker given π would never want to deviate from $BR(\pi)$, and similarly, by definition the victim would never want to deviate from its defense policy when assuming the worst possible attack. Thus, we consider such attack and defense policies as truly *optimal*. However, as the victim faces partial observability, an optimal defense for the victim is history-dependent in general. Consequently, the attacker's best response must also be history-dependent. Thus, Π and N consist of history-dependent policies in the definition above.

Although optimal Stackelberg strategies for Stochastic games are generally difficult to compute [68], we can exploit the special structure of the victim-attacker-M interaction to develop useful algorithms. Recall that at subtime t_2 in Algorithm 1, the attacker changes the observation to $o^\dagger$, and then the victim chooses an action $a = \pi(o^\dagger)$. If we simply give the victim the autonomy to choose any action a at this point rather than according to a fixed policy π, then this interaction evolves like a turn-based game. In fact, we show this game can be modeled as a partially observable turn-based stochastic game (POTBSG) [133]. POTBSGs exhibit much more structure than a general imperfect-information stochastic game, so enable more

efficient solution methods. We see the construction is almost identical to Definition 2.

Definition 4. The victim-attacker's POTBSG is $\overline{G} = (\bar{\mathcal{S}}_1 \cup \bar{\mathcal{S}}_2, \overline{\mathcal{O}}, \bar{\mathcal{A}}, \bar{P}, \bar{r}, \overline{\gamma}, \bar{\mu})$ where,

- $\bar{\mathcal{S}}_1 := \mathcal{S} \times \mathcal{O} \times \{\varnothing\}$ and $\bar{\mathcal{S}}_2 := \mathcal{S} \cup (\mathcal{S} \times \mathcal{O}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A} \times \mathcal{R})$.
- $\overline{\mathcal{O}}(\bar{s}) := o$ for $\bar{s} = (s, o, \varnothing)$ and $\overline{\mathcal{O}}(\bar{s}) := \bar{s}$ otherwise.
- $\bar{\mathcal{A}}(s) := \mathcal{B}(s)$, $\bar{\mathcal{A}}(s, o) := \mathcal{B}(s, o)$, $\bar{\mathcal{A}}(s, o, \varnothing) := \mathcal{A}$, $\bar{\mathcal{A}}(s, o, a) := \mathcal{B}(s, o, a)$, and $\bar{\mathcal{A}}(s, o, a, r) := \mathcal{B}(s, o, a, r)$.
- Let $\bar{s} \in \bar{\mathcal{S}}$, $\bar{a} \in \bar{\mathcal{A}}(\bar{s})$, and $\bar{s}' \in \bar{\mathcal{S}}$.

  1. If $\bar{s} = s$, then $\bar{a} = s^\dagger$ and $\bar{s}' = (s^\dagger, o)$:
     $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := \mathcal{O}(o \mid s^\dagger)$.
  2. If $\bar{s} = (s, o)$, then $\bar{a} = o^\dagger$ and $\bar{s}' = (s, o^\dagger, \varnothing)$:
     $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := \pi(a \mid o^\dagger)$.
  3. If $\bar{s} = (s, o, \varnothing)$, then $\bar{a} = a$ and $\bar{s}' = (s, o^\dagger, a)$:
     $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := 1$.
  4. If $\bar{s} = (s, o, a)$, then $\bar{a} = a^\dagger$ and $\bar{s}' = (s, o, a^\dagger, r)$:
     $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := R(r \mid s, a^\dagger)$.
  5. If $\bar{s} = (s, o, a, r)$, then $\bar{a} = r^\dagger$ and $\bar{s}' = s'$:
     $\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := P(s' \mid s, a)$.

  All other transitions have probability 0.

- Let $\bar{s} \in \bar{\mathcal{S}}$, and $\bar{a} \in \bar{\mathcal{A}}(\bar{s})$. $\bar{r}_1(\bar{s}, \bar{a}) := r^\dagger$ and $\bar{r}_2(\bar{s}, \bar{a}) := g(s, a, r^\dagger)$ if $\bar{s} = (s, o, a, r)$ and $\bar{r}_1(\bar{s}, \bar{a}) := \bar{r}_2(\bar{s}, \bar{a}) := 0$ otherwise.
- $\overline{\gamma} := \gamma^{1/5}$.

- $\bar{\mu}(s) := \mu(s)$ for $s \in \mathcal{S}$ and $\bar{\mu}(\bar{s}) := 0$ otherwise.

Note that $\bar{\mathcal{S}}_1$ is the set of states in which the victim takes an action, and $\bar{\mathcal{S}}_2$ is the set of states in which the attacker takes an action. The observation and action set $\overline{\mathcal{O}}$ and $\bar{\mathcal{A}}$ as functions of the states are combined for the two players, and this implies that the observations and actions for the victim are $\bar{\mathcal{A}}(\bar{\mathcal{S}}_1)$ and $\overline{\mathcal{O}}(\bar{\mathcal{S}}_1)$, and for the attacker are $\bar{\mathcal{A}}(\bar{\mathcal{S}}_2)$ and $\overline{\mathcal{O}}(\bar{\mathcal{S}}_2)$. Observe that $V^{\pi,\nu}_{\mathcal{G},1} = V^{\pi,\nu}_1$ and $V^{\pi,\nu}_{\mathcal{G},2} = V^{\pi,\nu}_{\bar{M}}$ and so G is just the normal-form representation of the POTBSG $\mathcal{G}$.

Proposition 3. *Any WSE for $\mathcal{G}$ yields an optimal defense policy.*

In general, methods to compute WSE are unknown. However, we show many settings where a WSE for $\mathcal{G}$ can be computed, even efficiently. First, suppose the attacker is completely adversarial so that $\mathcal{G}$ becomes a zero-sum game. In this case, it is known that $WSE = SSE = NE$. Thus, it suffices to compute an NE for a zero-sum POTBSG.

Proposition 4. *If the attacker is completely adversarial, an optimal defense policy can be computed as an NE of $\mathcal{G}$ using any planning or distributed learning algorithms for zero-sum POTBSGs.*

Note, it is important that the victim uses a distributed learning algorithm since it would not be able to see the attacker's manipulations, only the effects of the manipulations, nor be able to collaborate with the attacker. From Proposition 3, we see that the victim can compute an optimal defense policy to an adversarial attacker by computing any CCE to $\mathcal{G}$. However, even computing an approximately optimal Markovian policy against a fixed attack is equivalent to solving a POMDP, which is NP-hard [73]. Thus, computing near-optimal defenses is intractable in the worst case.

Proposition 5. *For any $\epsilon > 0$ an ϵ-approximate optimal defense policy is NP-hard to compute even when restricting Π and N to be the class of Markovian policies.*

Efficient Methods. The main bottleneck to computing defenses efficiently in fully-observable systems is the presence of perceived-state attacks. Absent these attacks, the POTBSG specializes to a traditional TBSG, which is a special case of a stochastic game.

Observation 1. *When M is fully observable and the attacker cannot perform perceived-state attacks, $\mathcal{G}$ simplifies to a TBSG.*

In the adversarial case, we see that $\mathcal{G}$ is simply a zero-sum TBSG. In zero-sum TBSGs, even stationary NE can be computed or learned efficiently [27] unlike the case with CCE for MGs [29] and the solutions are exact.

Proposition 6. *If M is fully-observable, no perceived-state attacks are allowed, and M's rewards have finite support (or no reward attacks are allowed), and the attacker is adversarial, then an* optimal stationary *defense policy can be computed in polynomial time and learned with polynomial sample complexity.*

Although it is unclear whether Markovian policies guarantee the victim as much value as history-dependent ones, Markovian policies are commonplace since they are easier to store and deploy in practice. In fact, for the finite-horizon planning setting, the attacker need not be restricted. We give polynomial time planning algorithms to compute an optimal defense so long as perceived-state attacks are banned. To our knowledge, this is the first non-trivial setting for which WSE can be computed efficiently and the first non-trivial setting for which SSE can be computed beyond single-period games.

Theorem 2. *If M is fully-observable and has a finite horizon, no perceived-state attacks are allowed, and M's rewards have finite support (or no reward attacks are allowed), then an optimal defense policy can be computed in polynomial time.*

The intuition is the victim can simulate the attacker's best-response function using backward induction. Once it knows the best response for a particular stage game, it can then brute-force find the best action to take at that stage. The key insight is that the attacker's best response is always deterministic since it gets to see the victim's realized actions. Thus, the victim also has no benefit from randomization. As such, the victim can brute-force compute its optimal deterministic action to take during a single stage and then propagate that solution backward to be used in previous times.

To illustrate this, we derive a backward induction algorithm for efficient defense against action attacks and present the full defense algorithm in the Appendix. Suppose the victim has already committed to $\{\pi_t^*\}_{t=h+1}^{H}$, where H is the finite time-horizon. Clearly, for any choice of victim's action a, the attacker's best response to a and the future partial policy is:

$$BR_h(s,a) = \arg\max_{a^\dagger \in \bar{\mathcal{A}}(s,a)} g_h(s,a,r_h(s,a)) + \mathbb{E}_{s' \sim P_h(s,a^\dagger)} V^*_{h+1,2}(s', \pi^*_{h+1}(s')),$$

where $V^*_{h,2}(s,a)$ is the maximum value achieved. Then, the victim can compute its best action for the stage game (h,s) as a maximizer of,

$$V^*_{h,1}(s) = \max_{a \in \mathcal{A}} \min_{a^\dagger \in BR_h(s,a)} r_h(s,a^\dagger) + \mathbb{E}_{s' \sim P_h(s,a^\dagger)} V^*_{h+1,1}(s').$$

The construction for defending against all non-perceived state surfaces is a bit more complicated but retains this same structure.

Remark 3 (Multi-Agent Extension). We note that all of our results remain the same when multiple victims are present. This can be done without changing any of the previous notations by interpreting $\mathcal{A} = A_1 \times \ldots \times A_n$ as the joint action space and π as a joint policy. From the attacker's perspective, attacking many victims just looks like attacking a single victim with a large action space. A WSE in $\mathcal{G}$ still breaks up into an independent joint policy for the victims and the attacker, but the joint policy may require the victims to correlate with each other.

2.5 Experiments

We illustrate our frameworks with a classical grid-world shortest path problem with obstacles. Here, each state is a cell in a $n \times n$ grid. Some grid cells are filled with lava and so dangerous to the victim. From any cell, the victim can move left (L), right (R), up (U), or down (D) so long as it remains on the grid. In addition, the victim can stay (S) in its current cell. The agent wishes to get from the top-left cell $(0, 0)$ to the bottom-right, "goal", cell $(n-1, n-1)$ as quickly as possible while avoiding lava. To capture this goal, we assume the victim receives a reward of 1 for entering the goal cell and continues to receive a reward of 1 for each time it remains there to incentivize the victim to reach the goal quickly. We also assume the victim receives a penalty reward of $-H$ whenever it enters a lava cell, where H is the finite horizon.

Here, we test our methods on a 10×10 grid world with $H = 20$ so that the victim has enough time to reach the goal and stay there. We computed an optimal policy π^* for the grid, which achieves the victim a value of 3. In Figure 2.1 we visualize π^* through the path the victim follows when using π^*. The black cells represent a cell the victim entered during its interaction. The orange cells represent lava.

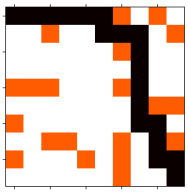

Figure 2.1: Optimal Policy Path.

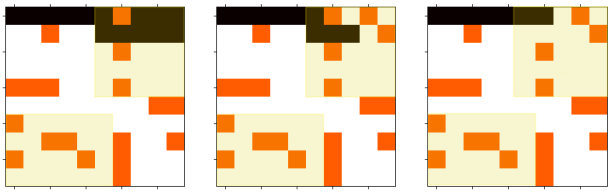

Figure 2.2: Attacked Paths.

2.5.1 Grid Attacks

The attacker can utilize its surfaces to disrupt the victim's path. For simplicity, assume that the attacker is purely adversarial and so it seeks to prevent the victim from reaching the goal and even trick it into lava cells if possible. Suppose that most of the grid is under security and so attacks cannot be safely made. The attacker is restricted to only attacking edges of the grid, which are not monitored. Here, the regions include the top-right subgrid and the bottom-left subgrid shaded in yellow. However, in those regions, it may use any attack it likes from its given surface.

In Figure 2.2, we see from left to right the path under an optimal perceived-state attack, true-state attack, and action attack. The agent receives -100, 0, and -160 value from each attack respectively. In all cases, the victim no longer reaches the goal after getting attacked in the top-right subgrid. We see the perceived-state attack functions by tricking the agent into entering lava; whereas the action attack simply forces the victim into lava. On the other hand, the state attacks can transport the victim into lava, but they immediately leave and so suffers less damage than in the other attacks despite seeming to be the most powerful.

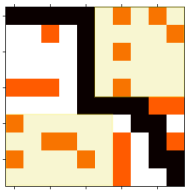

Figure 2.3: Defense Policy Path

2.5.2 Grid Defense

We see that if the victim simply follows π^*, the effects of attacks can be catastrophic. The victim knows the upper-right and bottom-left subgrids are not monitored and so can assume attacks are conducted there. Using this information, the defense algorithm yields a policy $\hat{\pi}$ that completely avoids the unsafe region. The victim still achieves the optimal value of 3 even under the strongest-possible attack. The new path under attack is illustrated in Figure 2.3. We see the robust path simply squeezes between the two unsafe regions.

2.6 Conclusion

In this chapter, we rigorously studied the attack and defense problems of reinforcement learning. We showed that for any attack's surface, a malicious attacker can optimally and efficiently maximize its own rewards by solving a higher lever meta-MDP. Even against linear environments, an attacker can still efficiently compute optimal attacks. Thus, we call for an agent to play a robust policy to be safe against such attacks. To this end, we formally defined the defense problem to be a weak-Stackelberg equilibrium of the natural partially-observable turn-based stochastic game that is induced by the victim-attacker-environment interaction. In the zero-sum setting, we showed the defense problem boils down to finding a Nash equilibrium in a zero-sum POTBSG so standard planning and learning methods can find optimal defense

policies. When perceived-state attacks are not allowed, the victim can also compute an optimal defense policy in polynomial time using a robust backward induction algorithm. Although we present an optimal defense, this defense may not be useful if the attacker is too powerful. It is critical for the victim to improve its detection abilities to restrict the attacker's feasible actions.

Chapter 3

Misinformation Attacks

Acknowledgments. This chapter was a joint work with Young Wu, Yudong Chen, Jerry Zhu, and Qiaomin Xie that appeared in RLC 2024.

Abstract. We study security threats to Markov games due to information asymmetry and misinformation. We consider an attacker player who can spread misinformation about its reward function to influence the robust victim player's behavior. Given a fixed fake reward function, we derive the victim's policy under worst-case rationality and present polynomial-time algorithms to compute the attacker's optimal worst-case policy based on linear programming and backward induction. Then, we provide an efficient inception ("planting an idea in someone's mind") attack algorithm to find the optimal fake reward function within a restricted set of reward functions with dominant strategies. Importantly, our methods exploit the universal assumption of rationality to compute attacks efficiently. Thus, our work exposes a security vulnerability arising from standard game assumptions under misinformation.

3.1 Introduction

As multi-agent systems become increasingly decentralized and privacy-focused, games with incomplete information become inevitable. In many scenarios, a player only has partial information about the opponent's rewards and rationality, gleaned from external sources like the internet. However, misinformation spread by the opponent—possibly through fake news—can significantly impact the player's decision-making. For example, participants in first-price auctions may intentionally misrepresent their intended bids to manipulate other bids downward. To build robust multi-agent systems, it is crucial to understand the impact of misinformation on games.

We focus on two-player Markov Games (MG). We suppose that the second player, the attacker, knows both reward functions, (R_1, R_2). In contrast, the first player, the victim, only knows its reward function, R_1, and a misinformed attacker reward function, $R_2^\dagger$. A robust victim also constructs an uncertainty set $\Pi_2^b(R_2^\dagger)$ of possible attacker policies. Nevertheless, the attacker can choose $R_2^\dagger$ to manipulate the victim's behavior. We call these fake rewards *inception attacks*. The attacker's goal is to design an inception attack that optimizes its worst-case utility.

Although inception attacks can be devastating, computing optimal attacks is often challenging. Unlike standard reward poisoning [120], an inception attack can not modify both players' rewards, which is necessary to illicit arbitrary victim behavior. Even if an oracle gave the attacker optimal fake rewards, computing a worst-case optimal attacker policy is a constrained optimization problem with nested maximins. Moreover, due to the information asymmetry, the attacker cannot utilize standard algorithms for computing robust optimization equilibrium (ROE) [2] or Bayes-Nash equilibrium (BNE) [50] to tackle this lower-level policy optimization problem.

Our Contributions. Although the computational complexity of inception might seem to limit its threat, we show that inception attacks can be efficiently computed by leveraging the universal rationality assumptions in multi-agent reinforcement learning (MARL). Specifically, for any rational or robust victim, we present an efficient algorithm for computing optimal dominant-policy inception attacks. The key insight is a rational victim always best-responds to a perceived attacker dominant strategy. Consequently, if the attacker focuses on fake reward functions admitting a dominant strategy, its complex optimization can be solved efficiently via backward induction. Our work exposes a security vulnerability arising from standard game assumptions under misinformation, motivating the need for novel approaches to building robust multi-agent systems.

To develop our inception algorithm, we first characterize outcomes in MGs with misinformation under worst-case rationality. Armed with these insights, we propose an efficient approach to compute the corresponding worst-case optimal policy for a given inception attack. Our method involves iteratively solving linear programs (LPs) based on worst-case Q functions. We derive these LPs by dualizing the best-response polytope, which transforms the maximin problems into maximization problems. Our approach accommodates any finitely generated victim uncertainty set, including completely naive and secure victims.

3.1.1 Related Work.

Information Asymmetry. Incomplete information games were first studied through the framework of Bayesian games [50, 51, 52] and with the solution concept being BNE. To address the high sensitivity of BNE to the player's beliefs [98, 59], the work [54] introduced a more robust equilibrium concept called ex-post equilibrium, which is a NE under all possible realizations of the uncertain parameters. Going

beyond the need for belief distributions, [2] introduced the notion of robust games with the solution concept being ROE. However, both the BNE and ROE approaches require non-trivial assumptions about the information structure, namely, an uncertainty parametrization or distributional assumption on the opponent's rewards. Thus, they do not apply to our setting where the victim knows nothing concrete about the attacker's true rewards.

Reward Poisoning Attacks. Most reward-poisoning attacks, for example, Ma et al. [74], Rakhsha et al. [92, 93], Rangi et al. [94], Zhang and Parkes [127], Zhang et al. [128] in the single-agent setting, and Wu et al. [122, 121, 119] in the multi-agent setting, focus on changing the victim's perceived rewards to induce negative behaviors rather than changing the victim's perception of the attacker's rewards. Unlike reward poisoning, which may not be possible in situations where the victim knows their preferences, inception attacks are more often possible since they fake the preferences of the attacker, which is usually not public information. Our setting also differs from past work by Gleave et al. [41], Guo et al. [46] on adversarial multi-agent reinforcement learning where an attacker is one of the agents (or controls one of the agents): they studied the problem in which an attacker modifies the action of an agent to influence the behavior of another agent (the victim).

3.1.2 Notations

We defer formal definitions of standard concepts in game theory to Appendix B.1.

Normal-form Games. Let $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{n\times m}$ denote the reward matrices for the victim and attacker, respectively. We represent a pure strategy by a one-hot vector, so $e_i \in \mathbb{R}^n$ corresponds to the victim's strategy i and $e_j \in \mathbb{R}^m$ the attacker's strategy j. Let $\Delta(k) := \left\{s \in [0,1]^k \mid \sum_{i=1}^k s_i = 1\right\}$ denote the set of mixed

strategies, where $s \in \Delta(k)$ corresponds to playing e_i with probability s_i.

Markov Games. A finite-horizon *Markov game* [100] is defined by a tuple $G = (S, \mathcal{A}, R, P, H, \mu)$ with state-space S, joint action space $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 = [n] \times [m]$ ($[i] := \{1, \ldots, i\}$), joint reward function R, transition function P, horizon H, and initial state distribution μ. We denote by $\pi = \{\pi_{1,h}(s) \in \Delta(n) \times \Delta(m)\}_{h,s}$ a joint Markovian policy. Let Π_i denote the set of all Markovian policies for player $i \in \{1, 2\}$ (victim and attacker). The value received by player i under π is the expected total rewards over H steps: $V_i^\pi := \mathbb{E}_G^\pi \left[\sum_{h=1}^H \pi_{1,h}(s_h)^\top R_{i,h}(s_h)\pi_{2,h}(s_h)\right]$. Similarly we define the stage value, $V_{i,h}^\pi(s)$, for each $h \in [H]$ by summing rewards over steps h through H. Throughout the paper, we assume that players know the transition function P.

3.2 Inception

Reward Uncertainty. We formalize misinformation threats through Markov games with reward uncertainty. Suppose that the victim has learned an alleged $R_2^\dagger$ directly from the attacker or external sources. A robust victim is aware that $R_2^\dagger$ may be inaccurate, so it constructs an uncertainty set $\mathcal{U}(R_2^\dagger)$ that it believes contains the attacker's true rewards. Furthermore, the victim believes the attacker behaves as playing some policy $\pi_2 \in \Pi_2^b(\mathcal{U}(R_2^\dagger))$, which depends on the belief rewards. To simplify notation, we assume the victim's belief about the attacker takes the form $\Pi_2^b(R_2^\dagger) \subseteq \Pi_2$, with the understanding that the victim may be using robust reasoning inside the belief function.

Assumption 1 (Victim's Belief)**.** The victim knows some uncertain reward function $R_2^\dagger$ and believes the attacker's policy must lie in the set $\Pi_2^b(R_2^\dagger)$. Furthermore, this is common knowledge.

Example 1 (Naive Belief)**.** If the victim believes it knows exactly which policy $\pi_2^\dagger$ the attacker will play, then $\Pi_2^b(R_2^\dagger) = \{\pi_2^\dagger\}$.

Example 2 (Secure Belief)**.** If the victim believes it knows nothing about the attacker, it may assume any attacker policy is possible, $\Pi_2^b(R_2^\dagger) = \Pi_2$.

Example 3 (Rational Belief)**.** If the victim believes the standard assumption of common-knowledge rationality, which is the case if it uses any standard MARL algorithm, then it assumes the attacker is rational. Concretely, the victim might assume the attacker plays some solution to the perceived game, $\Pi_2^b(R_2^\dagger) = \{\pi_2 \in \Pi_2 \mid \exists \pi_1 \in \Pi_1, (\pi_1, \pi_2) \in Sol(R_1, R_2^\dagger)\}$, where Sol is any standard solution concept such as DSE, NE, and maximin equilibrium[1]. In this work, we focus on inception attacks that only require the most basic form of rationality: rational agents never play strictly dominated strategies [120], which includes all the Sol options above.

3.2.1 Game Outcomes for Fixed $R_2^\dagger$

For any fixed $R_2^\dagger$, we can reason how both players will behave when the victim believes the attacker's policy is contained in the uncertainty set $\Pi_2^b(R_2^\dagger)$. To formally reason about the outcomes of such games, we turn to the standard notion of worst-case rationality [2].

Assumption 2. (Worst-Case Rationality) Both players seek to optimize their worst-case value given their available information.

Victim Behavior. For the victim to be robust, it should optimize against the worst possible policy the attacker could play. By Assumption 1, it need only consider attacker policies in $\Pi_2^b(R_2^\dagger)$.

[1]The assumption also holds for CCE, where Sol corresponds to the marginal policy of the CCE for each player.

Observation 2 (Victim Behaviour)**.** *Under Assumption 1 and Assumption 2, the victim plays some policy* $\pi_1^* \in \Pi_1^*(R_2^\dagger)$ *and achieves the optimal worst-case value* $V_1^*(R_2^\dagger)$ *where,*

$$\Pi_1^*(R_2^\dagger) := \arg\max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2^b(R_2^\dagger)} V_1^{\pi_1,\pi_2} \quad \textit{and} \quad V_1^*(R_2^\dagger) := \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2^b(R_2^\dagger)} V_1^{\pi_1,\pi_2}. \tag{VBR}$$

We observe that this behavior may be computationally intractable in general but is provably optimal under worst-case rationality. Also, this behavior can be viewed as a constrained security strategy that exploits the victim's beliefs to achieve better outcomes. This behavior directly generalizes security strategies, corresponding to the case when $\Pi_2^b(R_2^\dagger) = \Pi_2$.

Attacker Behavior. According to Assumption 1, the attacker knows $\Pi_2^b(R_2^\dagger)$. Thus, it can reason that the victim optimizes its worst-case value. Given this information, it can follow the same reasoning as the victim to predict how the victim behaves according to Observation 2. Specifically, the attacker should choose a policy that optimizes its value for the worst possible $\pi_1 \in \Pi_1^*(R_2^\dagger)$.

Observation 3. *Under Assumption 1 and Assumption 2, the attacker plays some* $\pi_2^* \in \Pi_2^*(R_2^\dagger)$ *and achieves the optimal worst-case value* $V_2^*(R_2^\dagger)$ *where,*

$$\Pi_2^*(R_2^\dagger) := \arg\max_{\pi_2 \in \Pi_2} \min_{\pi_1 \in \Pi_1^*(R_2^\dagger)} V_2^{\pi_1,\pi_2} \quad \textit{and } V_2^*(R_2^\dagger) := \max_{\pi_2 \in \Pi_2} \min_{\pi_1 \in \Pi_1^*(R_2^\dagger)} V_2^{\pi_1,\pi_2}. \tag{ABR}$$

Importantly, the attacker exploits its information asymmetry to constrain the inner minimization. This allows the attacker to achieve a higher value than it would from a standard security strategy.

Overall, we can see exactly how the Markov game with reward uncertainty will play out.

Proposition 7 (Game Outcomes)**.** *For any fixed* $R_2^\dagger$*, under Assumption 1 and Assumption 2,* (π_1^*, π_2^*) *is a solution to the game if and only if* $(\pi_1^*, \pi_2^*) \in \Pi_1^*(R_2^\dagger) \times \Pi_2^*(R_2^\dagger)$.

3.2.2 Inception Attacks

The attacker can induce the fake reward $R_2^\dagger$ that the victim learns, possibly by spreading misinformation. For any induced $R_2^\dagger$, the attacker can achieve up to $V_2^*(R_2^\dagger)$ value in the worst-case according to Observation 3. Thus, the attacker should choose an inception attack, $R_2^\dagger$, that maximizes $V_2^*(R_2^\dagger)$.

Definition 5 (Inception)**.** An *optimal inception attack* is any $R_2^\dagger$ that achieves V_2^* where,

$$V_2^* := \max_{R_2^\dagger} V_2^*(R_2^\dagger). \tag{INC}$$

In general, (INC) is a complex, bi-level optimization problem. However, this does not mean the victim is safe from such attacks. We show in Section 3.3 that damaging inception attacks can be computed in polynomial time for many settings.

Example 4 (Inception Attack)**.** Consider the simple normal-form game (R_1, R_2) and its corresponding inception-attack-induced game $(R_1, R_2^\dagger)$ given in Figure 3.1. Also, suppose that the victim believes the attacker plays its part of an NE for the faked game, i.e., $\Pi_2^b(R_2^\dagger) = \{y \mid \exists x,\ (x, y) \in \text{NE}(R_1, R_2^\dagger)\}$.

1. The original game in Figure 3.1a has a unique NE that is the pure strategy (D, L). Thus, $\Pi_2^b(R_2) = \{L\}$ and the victim plays its best-response D. This leads to the attacker always achieving a value of 0.

2. The fake game in Figure 3.1b has a unique NE which is the pure strategy (U, R). Thus, $\Pi_2^b(R_2^\dagger) = \{R\}$ and the victim plays its best-response U. This

leads to the attacker always achieving its highest possible value of 5 for the true game.

Therefore, the attacker can simply fake that it prefers action R while it actually prefers action L to manipulate the victim into achieving its ideal value.

	L	R
U	0, 5	1, 0
D	1, 0	0, 0

(a) True Game

	L	R
U	0, 5	1, 5+ϵ
D	1, 0	0, ϵ

(b) Inception Attack

Figure 3.1: Inception Example

3.3 Efficient Inception Algorithms

In this section, we show that for certain families of victims, the optimal inception attacks can be computed efficiently. To start, we show for a fixed $R_2^\dagger$ how the attacker can efficiently compute some best response policy in $\Pi_2^*(R_2^\dagger)$, which is already a complex problem. Then, we move on to computing optimal inception attacks for restricted classes of reward functions.

3.3.1 Efficiently Exploiting $R_2^\dagger$

Suppose that $R_2^\dagger$ is fixed. We observe that computing some $\pi_2 \in \Pi_2^*(R_2^\dagger)$ is a complicated optimization problem with constraints and a nested maximin optimization. Specifically,

$$\begin{aligned} \Pi_2^*(R_2^\dagger) = &\arg\max_{\pi_2^* \in \Pi_2} \min_{\pi_1^* \in \Pi_1^*} V_2^{\pi_1^*, \pi_2^*} \\ &\text{s.t. } \Pi_1^* = \arg\max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2^b(R_2^\dagger)} V_1^{\pi_1, \pi_2}. \end{aligned} \tag{3.1}$$

The optimization (3.1) can be arbitrarily complicated due to the arbitrary belief set $\Pi_2^b(R_2^\dagger)$. To have any hope of efficient solutions, we must restrict the belief set. Here, we consider any belief set that is a per-stage mixture of some finite set of base policies.

Assumption 3 (Finite Generation)**.** The victim's belief set is $\Pi_2^b(R_2^\dagger) = \Delta(\Pi)$, where $\Pi := \{\pi_2^1, \ldots, \pi_2^K\} \subseteq \Pi_2$ is a finite set of attacker policies and $\Delta(\Pi)$ is the simplex of per-stage mixings of Π, i.e.,

$$\Delta(\Pi) := \left\{ \pi \in \Pi_2 \mid \forall (h,s),\ \exists p \in \Delta(K) \ \text{ s.t. } \pi_{1,h}(s) = \sum_{k=1}^{K} p_k \pi_{2,h}^k(s) \right\}. \tag{3.2}$$

Normal-form Games

To see how Assumption Assumption 3 enables efficient computation, consider a normal-form game (A, B) and $\Pi = \{y_1, \ldots, y_K\} \subseteq \Delta(m)$.

Victim Best Response. It is well-known [28] that the victim can efficiently compute a maximin solution for A, i.e., $\max_{x \in \Delta(n)} \min_{y \in \Delta(m)} x^\top A y$, by solving the LP in Figure 3.2a. The inequalities $z \leq x^\top A e_j$ for all j ensure that x is the best response to any of the attacker's pure strategies, which then implies it is the best response to any mixture in $\Delta(m)$. In particular, x must be the best response to the worst possible mixed strategy in $\Delta(m)$.

The same reasoning applies if we replace each e_j with y_j. The inequalities $z \leq x^\top A y_j$ for all j then guarantee that x is a best response to the set $\Delta(\{y_1, \ldots, y_K\})$. Observe that we can equivalently formulate these inequalities by replacing A in Figure 3.2a with $A' := [Ay_1, \ldots, Ay_K] := A\Pi^\top$. Again, this implies x is the best response to the worst possible mixed strategy in $\Delta(\{y_1, \ldots, y_K\})$. Since $\Pi_1^*(R_2^\dagger)$ is the set of the victim's worst-case best responses to $\Pi_2^b(R_2^\dagger) = \Delta(\{y_1, \ldots, y_K\})$, we can compute some $x \in \Pi_1^*(R_2^\dagger)$ by solving LP Figure 3.2a with the modified reward

matrix A'.

$$\begin{aligned} \max_{x\in\mathbb{R}^n, z\in\mathbb{R}} \quad & z \\ \text{s.t.} \quad & z \le x^\top A e_j, \quad \forall j \in [m] \\ & 1^\top x = 1, \quad x \ge 0. \end{aligned}$$

(a) Victim's BR LP

$$\begin{aligned} \max_{y\in\mathbb{R}^m, w\in\mathbb{R}^K, \alpha\in\mathbb{R}} \quad & z^* 1^\top w - \alpha \\ \text{s.t.} \quad & \alpha + e_i^\top B y - e_i^\top A' w \ge 0 \quad \forall i \in [n] \\ & 1^\top y = 1, \quad y \ge 0 \quad w \ge 0. \end{aligned}$$

(b) Attacker's BR LP

Figure 3.2: Best-response LPs

Lemma 1. *If (x^*, z^*) is a solution to LP 3.2a for input $A' := [Ay_1, \ldots, Ay_K]$, then $V_1^*(R_2^\dagger) = z^*$ and $x^* \in \Pi_1^*(R_2^\dagger)$. Furthermore, $\Pi_1^*(R_2^\dagger) = \{x \in \Delta(n) \mid \forall j \in [K],\ x^\top A' e_j \ge z^*\}$ is a non-empty polytope.*

Attacker Best Response. Now that we have understood the victim's best response $\Pi_1^*(R_2^\dagger)$ polytope, the attacker can exploit this structure to compute some $y \in \Pi_2^*(R_2^\dagger)$. Recall the attacker's true reward matrix is B. For any fixed y, note that the attacker's inner minimization in (3.1) can be written as the following LP and its dual in Figure 3.3.

$$\begin{aligned} \min_{x\in\mathbb{R}^n_{\ge 0}} \quad & x^\top B y \\ \text{s.t.} \quad & z^* - x^\top A' e_j \le 0, \ \forall j \in [K], \\ & 1^\top x - 1 = 0. \end{aligned}$$

(a) Primal

$$\begin{aligned} \max_{w\in\mathbb{R}^K_{\ge 0}, \alpha\in\mathbb{R}} \quad & z^* 1^\top w - \alpha \\ \text{s.t.} \quad & \alpha + e_i^\top B y - e_i^\top A' w \ge 0, \ \forall i \in [n]. \end{aligned}$$

(b) Dual

Figure 3.3: Attacker's Inner Minimization

Applying $\max_{y\in\Delta(m)}$ on top of (3.3b) yields the LP in Figure 3.2b, which computes a $y \in \Pi_2^*(R_2^\dagger)$. We give the full derivation in the Appendix.

Lemma 2. *If (y^*, w^*, α^*) is a solution to LP 3.2b, then $V_2^*(R_2^\dagger) = z^* 1^\top w^* - \alpha^*$ and $y^* \in \Pi_2^*(R_2^\dagger)$. Furthermore, $\Pi_2^*(R_2^\dagger)$ is a non-empty polytope.*

Algorithm 2 Normal-Form Game Attacker Best Response

Input: Π, A, and B

$A' \leftarrow A\Pi^T$

$(x^*, z^*) \leftarrow Sol(LP\ 3.2a(A'))$

$(y^*, w^*, \alpha^*) \leftarrow Sol(LP\ 3.2b(z^*, A', B))$

return $(y^*, z^*, z^* 1^\top w^* - \alpha^*)$

Therefore, the attacker can compute a $y \in \Pi_2^*(R_2^\dagger)$ by first computing a solution (x^*, z^*) to LP 3.2a and then using z^* to formulate and solve LP 3.2b. Importantly, the attacker can solve LP 3.2a due to the information asymmetry: it knows the victim's A. The computation is summarized in Algorithm 2.

Theorem 3. *If $K \leq poly(m)$, then under Assumption 3 the attacker can compute some $y \in \Pi_2^*(R_2^\dagger)$ for a normal-form game in polynomial time by using Algorithm 2.*

Markov Games

To extend our results to full Markov games, we solve our LPs on each stage game via backward induction. To formalize this approach, we study the worst-case stage value and its corresponding worst-case Q functions:

$$V_{1,h}^*(s) := \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2^b(R_2^\dagger)} V_{1,h}^{\pi_1,\pi_2}(s) \text{ and } V_{2,h}^*(s) := \max_{\pi_2 \in \Pi_2} \min_{\pi_1 \in \Pi_1^*(R_2^\dagger)} V_{2,h}^{\pi_1,\pi_2}(s), \tag{3.3}$$

$$Q_{i,h}^*(s)[a_1, a_2] := R_{i,h}(s, a_1, a_2) + \sum_{s'} P_h(s' \mid s, a_1, a_2) V_{i,h+1}^*(s'). \tag{3.4}$$

In particular, for each $h \in [H], s \in S$, the worst-case stage-value functions $V_{i,h}^*(s)$ can be computed from the worst-case Q functions $Q_{i,h}^*(s)$, using Algorithm 2 with $(Q_{1,h}^*(s), Q_{2,h}^*(s))$ as the norm-form game reward matrix. We let $\pi_{1,h}(s) := \{\pi_{2,h}^1(s), \ldots, \pi_{2,h}^K(s)\}$.

Lemma 3. *For all h, s, we have that Algorithm 2$(\pi_{1,h}(s), Q_{1,h}^*(s), Q_{2,h}^*(s))$ outputs $(*, V_{1,h}^*(s), V_{2,h}^*(s))$.*

Algorithm 3 Markov Game Attacker Best Response

Input: Π and G

$V^*_{i,H+1}(s) = 0$ for all $s \in S$.
for $h = H$ down to 1 **do**
  for $s \in S$ **do**
    $Q^*_{1,h}(s), Q^*_{2,h}(s) \leftarrow$ Equation (3.4)
    $\pi^*_{2,h}(s), V^*_{1,h}(s), V^*_{2,h}(s) \leftarrow$ *Algorithm 2*$(\pi_{1,h}(s), Q^*_{1,h}(s), Q^*_{2,h}(s))$
return $\pi^*_2 := \{\pi^*_{2,h}(s)\}_{h,s}$

Since the worst-case value is uniquely defined, we can use backward induction to compute a solution for the whole Markov game in Algorithm 3.

Theorem 4. *If $K \leq poly(m)$, then under Assumption 3 the attacker can compute some $\pi_2 \in \Pi^*_2(R^\dagger_2)$ for a Markov game in polynomial time using Algorithm 3.*

Remark 4 (Secure Victims). If the victim does not trust $R^\dagger_2$ as in Example 2 and simply ignores the information by computing a maximin strategy, $\max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} V_1^{\pi_1,\pi_2}$, the attacker can still exploit its information asymmetry. In particular, it can compute its best response in polynomial time using Algorithm 3 on $\Pi = \{\pi^j_2\}_{j=1}^m$ where $\pi^j_{2,h}(s) := e_j$. This leads to $\Delta(\Pi) = \Pi_2$.

3.3.2 Efficiently Optimizing $R^\dagger_2$

In the previous section, we saw how to compute best-response policies for a class of beliefs of the victim. However, to compute an optimal inception attack, we require additional structure on how the victim maps rewards to belief sets.

Assumption 4 (Common Rationality)**.** If $\pi^\dagger_2$ is an ι-strictly dominant Markov-perfect strategy for $R^\dagger_2$, then $\Pi^b_2(R^\dagger_2) = \{\pi^\dagger_2\}$.

Remark 5. (Rationality) Note that Assumption 4 holds whenever the victim believes common knowledge rationality as in Example 3. We again emphasize this assumption

is made by all standard MARL algorithms as rationality is the basis of these game-theoretic approaches.

Policy Reduction. Observe that if $\Pi_2^b(R_2^\dagger) = \Pi_2^b(R_2^{\dagger\dagger})$, then $V_2^*(R_2^\dagger) = V_2^*(R_2^{\dagger\dagger})$. Consequently, whenever $\Pi_2^b(R_2^\dagger) = \{\pi_2^\dagger\}$, we see that $V_2^*(R_2^\dagger)$ is completely determined by $\pi_2^\dagger$ and not the specific structure of $R_2^\dagger$. Thus, with a slight abuse of notation, we can view V_2^* as a function of $\pi_2^\dagger$ by defining $V_2^*(\pi_2^\dagger) := V_2^*(R_2^\dagger)$ where $R_2^\dagger$ is any reward functions satisfying $\Pi_2^b(R_2^\dagger) = \{\pi_2^\dagger\}$. Overall, we can reduce the problem of finding fake rewards to the problem of finding a fake policy.

If $\Pi_2^b(R_2^\dagger) = \{\pi_2^\dagger\}$, then by definition $\Pi_1^*(R_2^\dagger) = \arg\max_{\pi_1 \in \Pi_1} V_1^{\pi_1, \pi_2^\dagger} =: BR(\pi_2^\dagger)$ is just the victim's traditional best response to $\pi_2^\dagger$. In addition, $V_2^*(\pi_2^\dagger) = \max_{\pi_2 \in \Pi_2} \min_{\pi_1 \in BR(\pi_2^\dagger)} V_2^{\pi_1, \pi_2}$ can be efficiently computed using Algorithm 3. As only deterministic policies can be dominant, this simplifies the attacker's search to a finite set. Thus, the policy version of the problem is simpler to tackle. The attacker can then do inverse reward engineering to find a reward function for which $\pi_2^\dagger$ is a dominant strategy, which is possible even for robust victims [120].

Lemma 4 (Reward-Policy Reduction)**.** *Under Assumption 4,*

$$\max_{R_2^\dagger \in \mathcal{D}} V_2^*(R_2^\dagger) = \max_{\pi_2^\dagger \in \Pi_2^D} V_2^*(\pi_2^\dagger), \tag{3.5}$$

where $\mathcal{D}$ is the set reward functions with an ι-strictly dominant Markov-perfect strategy, and Π_2^D is the set of deterministic attacker policies. We let $\hat{V}_2 := \max_{\pi_2^\dagger \in \Pi_2^D} V_2^*(\pi_2^\dagger)$ denote the optimal value.

Lemma 4 states that if the misinformation-induced reward function $R_2^\dagger$ is restricted to the set admitting strictly dominant strategies, one can solve the optimal inception attack problem by solving the pure strategy optimization problem. We note this

restricted set is infinite and captures many interesting reward functions.

Remark 6 (Reward Design). We note the choice of $R^{\dagger}_{2,h}(s,a) = \iota\frac{(H-h+1)(H-h+2)}{2}\mathbb{I}[a_2 = \pi^{\dagger}_{2,h}(s)]$ suffices to ensure $\pi^{\dagger}_2$ is the dominant strategy in any stage game and can be computed in polynomial time. If there are other constraints on the reward function, other reward poisoning frameworks can be used black box to compute optimal attacks.

Algorithmic Approach. For the normal-form game (A, B), it is easy to see that for any pure strategy $j \in [m]$ that $V^*_2(j) = \max_{y\in\Delta(m)} \min_{x\in BR(j)} x^\top By$ can be computed using *Algorithm* 2$(\{j\}, A, B)$ in polynomial time. The maximal pure strategy can then be found efficiently by iterating over all $j \in [m]$: $\hat{V}_2 = \max_j V^*_2(j)$. Thus, we can solve the policy problem for a normal-form game efficiently by repeatedly applying Algorithm 2.

This line of argument can be extended to Markov games by replacing (A, B) with the Q-function matrices and using backward induction. Suppose the attacker has already constructed a partial policy $\pi^{\dagger}_2$ for times $h+1, \ldots, H$. At time h and state s, the attacker can tentatively define $\pi^{\dagger}_{2,h}(s) = j$. For this choice, the attacker can reason about the victim's best-response set and value $\hat{V}_{1,h}(s, j)$, which is also constructed via backward induction. The attacker can then just choose the optimal j that leads to its highest worst-case stage value, $\hat{V}_{2,h}(s, j)$. Formally, we define,

$$\hat{V}_{2,h}(s) = \max_{\pi^{\dagger}_2\in\Pi^D_2} \min_{\pi_1\in BR(\pi^{\dagger}_2)} V^{\pi_1,\pi_2}_{2,h}(s) \text{ and } \hat{V}_{1,h}(s) = \max_{\pi_1\in\Pi_1} V^{\pi_1,\pi^{\dagger}_2}_{1,h}(s), \tag{3.6}$$

to be the value of the best inception policy for the attacker at the current stage and the victim's best response value to a fixed inception policy $\pi^{\dagger}_2$, respectively. We can similarly define the corresponding $\hat{Q}$ function through (3.4) by replacing V^* with $\hat{V}$.

Algorithm 4 Policy Inception

Input: Π and G

$\hat{V}_{i,H+1}(s) = 0$ for all $s \in S$.
for $h = H$ down to 1 **do**
  for $s \in S$ **do**
    $\hat{Q}_{1,h}(s), \hat{Q}_{2,h}(s) \leftarrow$ Equation (3.4)
    for $j \in [m]$ **do**
      $\pi^*_{2,h}(s), \hat{V}_{1,h}(s,j), \hat{V}_{2,h}(s,j) \leftarrow$ *Algorithm 2*$(\{j\}, \hat{Q}_{1,h}(s), \hat{Q}_{2,h}(s))$
    $\pi^\dagger_{2,h}(s) \leftarrow \arg\max_{j\in[m]} \hat{V}_{2,h}(s,j)$
    $\hat{V}_{i,h}(s) \leftarrow \hat{V}_{i,h}(s, \pi^\dagger_{2,h}(s))$ for $i \in [2]$
return $\pi^\dagger_2 := \{\pi^\dagger_{2,h}(s)\}_{h,s}$

Then, for any fixed $j \in [m]$, we define,

$$\hat{V}_{2,h}(s,j) = \max_{y\in\Delta(m)} \min_{x\in BR(j)} x^\top \hat{Q}_{2,h}(s) y \quad \text{and} \quad \hat{V}_{1,h}(s,j) = \max_{x\in\Delta(n)} x^\top \hat{Q}_{1,h}(s) e_j, \tag{3.7}$$

as the value when the attacker chooses $\pi^\dagger_{2,h}(s) = j$ at step h, and applies the optimal inception policy for times $h+1, \ldots, H$.

Lemma 5. *For all h, s, j, we have that Algorithm 2$(\{j\}, \hat{Q}_{1,h}(s), \hat{Q}_{2,h}(s))$ outputs $(*, \hat{V}_{1,h}(s,j), \hat{V}_{2,h}(s,j))$. Furthermore, if $j^* \in \arg\max_{j\in[m]} \hat{V}_{2,h}(s,j)$, then $\hat{V}_{i,h}(s) = \hat{V}_{i,h}(s,j^*)$ for each $i \in \{1,2\}$.*

In the same spirit as Algorithm 3, we can compute an optimal $\pi^\dagger_2$ using Algorithm 4.

Theorem 5. *Under Assumption 4, Algorithm 4 computes a fake policy achieving value $\hat{V}_2$ in polynomial time.*

Remark 7 (Dominant Mixtures). The algorithm can be extended to allow a mixture of a set of policies by changing $\{j\}$ to a subset of actions. This captures reward matrices with several equally dominant columns.

3.4 Conclusion

In this chapter, we studied misinformation attacks on two-player MGs. When the victim player only knows a false attacker reward function, we showed how the game plays out under worst-case rationality. Then, we showed how the attacker can compute its worst-case optimal policy in polynomial time. Using this method as a subroutine, the attacker can exploit the universal assumption of rationality in MARL to compute an optimal dominant-policy inception attack in polynomial time. Our work highlights that the standard rationality notions produce vulnerabilities when misinformation is present. Thus, new approaches are needed to build multi-agent systems that are robust against misinformation.

Chapter 4

Anytime Constraints

Acknowledgments. This chapter was a joint work with Jerry Zhu that appeared in AISTATS 2024.

Abstract. In this chapter, we introduce and study constrained Markov Decision Processes (cMDPs) with anytime constraints. An anytime constraint requires the agent to never violate its budget at any point in time, almost surely. Although Markovian policies are no longer sufficient, we show that there exist optimal deterministic policies augmented with cumulative costs. In fact, we present a fixed-parameter tractable reduction from anytime-constrained cMDPs to unconstrained MDPs. Our reduction yields planning and learning algorithms that are time and sample-efficient for tabular cMDPs so long as the precision of the costs is logarithmic in the size of the cMDP. However, we also show that computing non-trivial approximately optimal policies is NP-hard in general. To circumvent this bottleneck, we design provable approximation algorithms that efficiently compute or learn an arbitrarily accurate approximately feasible policy with optimal value so long as the maximum supported cost is bounded by a polynomial in the cMDP or the absolute budget. Given our hardness results, our approximation guarantees are the best possible under worst-case

analysis.

4.1 Introduction

Suppose M is a constrained Markov Decision Process (cMDP). An *anytime constraint* requires that cost accumulated by the agent's policy π is within the budget at any time, almost surely: $\mathbb{P}^{\pi}_{M}\left[\forall k \in [H],\ \sum_{t=1}^{k} c_t \leq B\right] = 1$. If Π_M denotes the set of policies that respect the anytime constraints, then a solution to the anytime-constrained cMDP is a policy $\pi^* \in \arg\max_{\pi \in \Pi_M} V^{\pi}_{M}$. For example, consider planning a minimum-time route for an autonomous vehicle to travel from one city to another. Besides time, there are other important considerations including [86] (1) the route does not exhaust the vehicle's fuel, and (2) the route is safe. We can model (1) by defining the fuel consumed traveling a road to be its cost and the tank capacity to be the budget. Refueling stations are captured using negative costs. We can model many safety considerations (2) similarly.

Since nearly every modern system is constrained in some way, one key step to modeling more realistic domains with MDPs is allowing constraints. To address this, a rich literature of constrained reinforcement learning (CRL) has been developed, almost exclusively focusing on expectation constraints [5, 1, 14] or high-probability (chance) constraints [19, 84, 23]. However, in many applications, especially where safety is concerned or resources are consumed, anytime constraints are more natural. An agent would not be reassured by the overall expected or probable safety of a policy when it is faced with a reality or intermediate time where it is harmed. For instance, a car cannot run on expected or future gas, so must satisfy the B budget at any time along the route. Similarly, in goal-directed RL [57, 80, 11], the goal must be achieved; maximizing reward is only a secondary concern. These issues are especially

crucial in medical applications [26, 87, 64], disaster relief scenarios [35, 117, 109], and resource management [75, 70, 90, 13].

Anytime constraints are natural, but introduce a plethora of new challenges. Traditional Markovian and history-dependent policies are rarely feasible and can be arbitrarily suboptimal; the cost history must also be considered. Naively using backward induction to compute an optimal cost-history-dependent policy is possible in principle for tabular cost distributions, but the time needed to compute the policy and the memory needed to store the policy would be super-exponential. Since the optimal solution value is a discontinuous function of the costs, using standard CRL approaches like linear programming is also impossible. In fact, not only is computing an optimal policy NP-hard but computing any policy whose value is approximately optimal is also NP-hard when at least two constraints are present.

Known works fail to solve anytime-constrained cMDPs. Expectation-constrained approaches [5, 89, 31, 61, 34] and chance-constrained approaches [84, 123, 23, 82] yield policies that arbitrarily violate an anytime constraint. This observation extends to nearly every known setting: Knapsack constraints [17, 22, 20], risk constraints [15, 23], risk sensitivity [126, 57, 102, 80], quantile constraints [60, 124], and instantaneous constraints [69, 37, 43]. If we were instead to use these models with a smaller budget to ensure feasibility, the resultant policy could be arbitrarily suboptimal if any policy would be produced at all. Dangerous-state [96, 107] and almost-sure [18] constraints can be seen as a special case of our model with binary costs. However, their techniques do not generalize to our more complex setting. Moreover, absent expectation constraints, none of these approaches are known to admit polynomial time planning or learning algorithms.

Our Contributions. We present the first formal study of anytime-constrained cMDPs. Although traditional policies do not suffice, we show that deterministic augmented policies are always optimal. In fact, an optimal policy can be computed by solving an unconstrained, augmented MDP using any standard RL planning or learning algorithm. Using the intuition of safe exploration and an atypical forward induction, we derive an augmented state space rich enough to capture optimal policies without being prohibitively large. To understand the resultant augmented policies, we design new machinery requiring a combination of backward and forward induction to argue about optimality and feasibility. Overall, we show our reduction to standard RL is *fixed-parameter tractable* (FPT) [33] in the cost precision when the cMDP and cost distribution are tabular. In particular, as long as the cost precision is logarithmic in the size of the cMDP, our planning (learning) algorithms are polynomial time (sample complexity), and the produced optimal policy can be stored with polynomial space.

Since we show computing any non-trivial approximately-optimal policy is NP-hard, we turn to approximate feasibility for the general case. For any $\epsilon > 0$, we consider additive and relative approximate policies that accumulate cost at most $B + \epsilon$ and $B(1 + \epsilon)$ anytime, respectively. [1] Rather than consider every cumulative cost induced by safe exploration, our approximation scheme inductively accumulates and projects the costs onto a smaller space. Following the principle of optimism, the approximate cost is constructed to be an underestimate to guarantee optimal value. Our approach yields planning (learning) algorithms that produce optimal value, and approximately feasible policies in polynomial time (sample complexity) for any, possibly non-tabular, cost distribution whose maximum supported cost is bounded by a polynomial in the cMDP or by the absolute budget. Given our hardness results,

[1] Critically, we can also ensure strict budget B feasibility but with a weaker value guarantee. See section 4.4.1.

this is the best possible approximation guarantee one could hope for under worst-case analysis. We also extend our methods to handle different budgets per time, general almost-sure constraints, and infinite discounting.

4.1.1 Related Work.

Knapsack Constraints. The knapsack-constrained frameworks [17, 22, 20] were developed to capture constraints on the learning process similar to bandits with knapsacks [6]. Brantley et al. [17] and Cheung [22] both constrain the total cost violation that can be produced during training. On the other hand, Chen et al. [20] introduces the RLwK framework that constrains the total cost used per episode. In RLwK, each episode terminates when the agent violates the budget for that episode. In all of these models, the environment is given the ability to terminate the process early; the final policy produced after learning need not satisfy any kind of constraint. In fact, the agent still keeps the reward it accumulated before violation, the agent is incentivized to choose unsafe actions in order to maximize its reward. Thus, such methods produce infeasible policies for our anytime constraints regardless of the budget they are given.

Almost Sure Constraints. Performing RL while avoiding dangerous states [96, 107, 44] can be seen as a special case of both anytime and expectation constraints with binary costs and budget 0. However, these works require non-trivial assumptions, and being a special case of expectation constraints implies their techniques cannot solve our general setting. Similarly, Castellano et al. [18] introduced almost sure constraints with binary costs, which can be seen as a special case of anytime constraints. However, they focus on computing minimal budgets, which need not lead to efficient solutions in general since the problem is NP-hard even with a budget of 1. Lastly, the infinite

time-average case with almost sure constraints has been thoroughly studied [97]. Since we focus on finite-horizon and discounted settings, our policies would always have a time-average cost of 0 and so those methods cannot produce policies that are feasible for anytime constraints.

4.2 Anytime Constraints

Constrained Markov Decision Processes. A (tabular, finite-horizon) Constrained Markov Decision Process is a tuple $M = (\mathcal{S}, \mathcal{A}, P, R, H, s_0, C, B)$, where (i) $\mathcal{S}$ is a finite set of states, (ii) $\mathcal{A}$ is a finite set of actions, (iii) $P_h(s, a) \in \Delta(S)$ is the transition distribution, (iv) $R_h(s, a) \in \Delta(\mathbb{R})$ is the reward distribution, (v) $H \in \mathbb{N}$ is the finite time horizon, (vi) $s_0 \in \mathcal{S}$ is the initial state, (vii) $C_h(s, a) \in \Delta(\mathbb{R}^d)$ is the cost distribution, and (viii) $B \in \mathbb{R}^d$ is the budget vector. Here, $d \in \mathbb{N}$ denotes the number of constraints. We overload notation by letting $C_h(s, a)$ denote both the cost distribution and its support. We also let $r_h(s, a) = \mathbb{E}[R_h(s, a)]$ denote the expected reward. Lastly, we let $S := |\mathcal{S}|$, $A := |\mathcal{A}|$, $[H] := \{1, \ldots, H\}$, and $|M|$ be the description size of the cMDP.

Interaction Protocol. A complete history with costs takes the form $\tau = (s_1, a_1, c_1, \ldots, s_H, a_H, c_H, s_{H+1})$, where $s_h \in \mathcal{S}$ denotes M's state at time h, $a_h \in \mathcal{A}$ denotes the agent's chosen action at time h, and $c_h \in C_h(s_h, a_h)$ denotes the cost incurred at time h. We let $\bar{c}_h := \sum_{t=1}^{h-1} c_t$ denote the cumulative cost up to (but not including) time h. Also, we denote by $\tau_h = (s_1, a_1, c_1, \ldots, s_h)$ the partial history up to time h and denote by $\mathcal{H}_h$ the set of partial histories up to time h. The agent interacts with M using a policy $\pi = (\pi_h)_{h=1}^H$, where $\pi_h : \mathcal{H}_h \to \Delta(\mathcal{A})$ specifies how the agent chooses actions at time h given a partial history.

The agent starts at state s_0 with partial history $\tau_1 = (s_0)$. For any $h \in [H]$, the

agent chooses an action $a_h \sim \pi_h(\tau_h)$. Afterward, the agent receives reward $r_h \sim R_h(s_h, a_h)$ and cost $c_h \sim C_h(s_h, a_h)$. Then, M transitions to state $s_{h+1} \sim P_h(s_h, a_h)$ and the history is updated to $\tau_{h+1} = (\tau_h, a_h, c_h, s_{h+1})$. This process is repeated for H steps total; the interaction ends once s_{H+1} is reached.

Objective. The agent's goal is to compute a π^* that is a solution to the following optimization problem4:

$$\begin{aligned} &\max_{\pi} \mathbb{E}_M^{\pi} \left[\sum_{h=1}^{H} r_h(s_h, a_h) \right] \\ &\text{s.t. } \mathbb{P}_M^{\pi} \left[\forall t \in [H],\ \sum_{h=1}^{t} c_h \leq B \right] = 1. \end{aligned} \tag{ANY}$$

Here, $\mathbb{P}_M^{\pi}$ denotes the probability law over histories induced from the interaction of π with M, and $\mathbb{E}_M^{\pi}$ denotes the expectation with respect to this law. We let $V^{\pi} := \mathbb{E}_M^{\pi} \left[\sum_{t=1}^{H} r_t(s_t, a_t) \right]$ denote the value of a policy π, $\Pi_M := \left\{ \pi \mid \mathbb{P}_M^{\pi} \left[\forall k \in [H],\ \sum_{t=1}^{k} c_t \leq B \right] = 1 \right\}$ denote the set of feasible policies, and $V^* := \max_{\pi \in \Pi_M} V^{\pi}$ denote the optimal solution value [2]. If there are no feasible policies, $V^* := -\infty$ by convention.

Optimal Solutions. It is well-known that expectation-constrained cMDPs always admit a randomized Markovian policy [5]. However, under anytime constraints, feasible policies that do not remember the cumulative cost can be arbitrarily suboptimal. The intuition is that without knowing the cumulative cost, a policy must either play it too safe and suffer small value or risk an action that violates the constraint.

Proposition 8. *Any class of policies that excludes the full cost history is suboptimal for anytime-constrained cMDPs. In particular, Markovian policies can be arbitrarily*

[2] By using a negative budget, we capture the covering constraints that commonly appear in goal-directed problems. We consider the other variations of the problem in the Appendix.

suboptimal even for cMDPs with $S = 1$ and $A = H = 2$.

Corollary 1. *(Approximately) optimal policies for cMDPs with expectation constraints, chance constraints, or their variants can arbitrarily violate an anytime constraint. Furthermore, (approximately) optimal policies for a cMDP defined by a smaller budget to achieve feasibility can be arbitrarily suboptimal.*

Although using past frameworks out of the box does not suffice, one might be tempted to use standard cMDP techniques, such as linear programming, to solve anytime-constrained problems. However, continuous optimization techniques fail since the optimal anytime-constrained value is discontinuous in the costs and budgets. Even a slight change to the cost or budget can lead to a dramatically smaller solution value.

Proposition 9. *V^* is a continuous function of the rewards, but a discontinuous function of the costs and budgets.*

Intractability. In fact, solving anytime-constrained cMDPs is fundamentally harder than expectation-constrained cMDPs; solving (ANY) is NP-hard. The intuition is that anytime constraints capture the knapsack problem. With a single state, we can let the reward at time i be item i's value and the cost at time i be item i's weight. Any deterministic policy corresponds to choosing certain items to add to the knapsack, and a feasible policy ensures the set of items fit in the knapsack. Thus, an optimal deterministic policy for the cMDP corresponds to an optimal knapsack solution.

On the other hand, a randomized policy does not necessarily yield a solution to the knapsack problem. However, we can show that any randomized policy can be derandomized into a deterministic policy with the same cost and at least the same value. The derandomization can be performed by inductively choosing any

supported action that leads to the largest value. This is a significant advantage over expectation and chance constraints which typically require stochastic policies.

Lemma 6 (Derandomization)**.** *For any randomized policy $\bar{\pi}$, there exists a deterministic policy π whose cumulative cost is at most $\bar{\pi}$'s anytime and whose value satisfies $V^{\pi} \geq V^{\bar{\pi}}$.*

Since the existence of a randomized solution implies the existence of a deterministic solution with the same value via Lemma 6, the existence of a high-value policy for an anytime-constrained cMDP corresponds to the existence of a high-value knapsack solution. Thus, anytime constraints can capture the knapsack problem. Our problem remains hard even if we restrict to the very specialized class of deterministic, non-adaptive (state-agnostic) policies, which are mappings from time steps to actions: $\pi : [H] \to \mathcal{A}$.

Theorem 6 (Hardness)**.** *Solving* (ANY) *is NP-complete even when $S = 1$, $A = 2$, and both the costs and rewards are deterministic, non-negative integers. This remains true even if restricted to the class of non-adaptive policies. Hardness also holds for stationary cMDPs so long as $S \geq H$.*

Given the hardness results in Theorem 6, it is natural to turn to approximation algorithms to find policies efficiently. The most natural approach would be to settle for a feasible, although, approximately-optimal policy. Unfortunately, even with only $d = 2$ constraints, it is intractable to compute a feasible policy with any non-trivial approximation factor. This also means designing an algorithm whose complexity is polynomial in d is likely impossible.

Theorem 7 (Hardness of Approximation)**.** *For $d \geq 2$, computing a feasible solution to* (ANY) *is NP-hard. Furthermore, for any $\epsilon > 0$, it is NP-hard to compute a feasible policy π satisfying either $V^{\pi} \geq V^{*} - \epsilon$ or $V^{\pi} \geq V^{*}(1 - \epsilon)$.*

Remark 8. Note, Theorem 7 does not only rule out the existence of fully-polynomial-time approximation schemes (FPTAS). Since $\epsilon > 0$ is arbitrary, it rules out any non-trivial approximation similar to the (non-metric) Traveling Salesman Problem.

4.3 FPT Reduction

Despite our strong hardness results, Theorem 6 and Theorem 7, we show for a large class of cMDPs, (ANY) can be solved efficiently. The key is to augment the state space of the system to capture the constraint consideration. In this section, we assume the cost distributions have finite support; we generalize to broader classes of distributions in Section 4.4.

Assumption 5. $n := \sup_{h,s,a} |C_h(s,a)| < \infty$.

Proposition 8 illustrates that a key issue with standard policies is that they cannot adapt to the costs seen so far. This forces the policies to be overly conservative or to risk violating the budget. At the same time, cost-history-dependent policies are undesirable as they are computationally expensive to construct and store in memory.

Instead, we claim the agent can exploit a sufficient statistic of the cost sequence: the *cumulative cost*. By incorporating cumulative costs carefully, the agent can simulate an unconstrained MDP, $\bar{M}$, whose optimal policies are solutions to (ANY). The main challenge is defining the augmented states, $\bar{\mathcal{S}}_h$.

Augmented States. We could simply define $\bar{\mathcal{S}}_h$ to be $\mathcal{S} \times \mathbb{R}^d$, but this would result in an infinite state MDP with a discontinuous reward function, which cannot easily be solved. The ideal choice would be $\mathcal{F}_h := \{(s, \bar{c}) \in \mathcal{S} \times \mathbb{R}^d \mid \exists \pi \in \Pi_M,\ \mathbb{P}_M^\pi [s_h = s, \bar{c}_h = \bar{c}] > 0\}$, which is the minimum set containing all (state, cost)-pairs induced by feasible policies. However, $\mathcal{F}_h$ is difficult to characterize.

Instead, we consider a relaxation stemming from the idea of *safe exploration*. Namely, we look at the set of all (state, cost)-pairs that the agent could induce if it repeatedly interacted with M and only took actions that would not violate the constraint given the current history. This set can be constructed inductively. First, the agent starts with $(s_0, 0)$ because it has yet to incur any costs. Then, if at time h, the agent has safely arrived at the pair $(s, \bar{c})$, the agent can now safely choose any action a for which $\Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1$.

Definition 6 (Augmented States)**.** $\bar{\mathcal{S}}_1 := \{(s_0, 0)\}$, and for any $h \geq 1$,

$$\bar{\mathcal{S}}_{h+1} := \Big\{(s', \bar{c}') \mid \exists (s, \bar{c}) \in \bar{\mathcal{S}}_h, a \in \mathcal{A}, c' \in C_h(s, a),$$
$$\bar{c}' = \bar{c} + c', \ \Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1, \ P_h(s' \mid s, a) > 0\Big\}.$$

Unlike the backward induction approaches commonly used in MDP theory, observe that $\bar{\mathcal{S}}$ is constructed using *forward induction*. This feature is critical to computing a small, finite augmented-state space. We also point out that $\bar{\mathcal{S}}_h$ is a relaxation of $\mathcal{F}_h$ since actions chosen based on past costs without considering the future may not result in a fully feasible path. Nevertheless, the relaxation is not too weak; whenever 0 cost actions are always available, $\bar{\mathcal{S}}_h$ exactly matches $\mathcal{F}_h$.

Lemma 7. *$\forall h \in [H+1]$, $\bar{\mathcal{S}}_h \supseteq \mathcal{F}_h$ and $|\bar{\mathcal{S}}_h| < \infty$. Furthermore, equality holds if $\forall h, s, \exists a$ for which $C_h(s, a) = \{0\}$.*

If the agent records its cumulative costs and always takes safe actions, the interaction evolves according to the following unconstrained MDP.

Definition 7 (Augmented MDP)**.** The *augmented MPD* $\bar{M} := (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{P}, \overline{R}, H, \bar{s}_0)$ where,

- $\bar{\mathcal{S}}_h$ is defined in Definition 6.

Algorithm 5 Reduction to Unconstrained RL

Input: cMDP M

$\bar{M} \leftarrow$ Definition 7(M)
$\pi, \mathcal{V}^* \leftarrow \text{Solve}(\bar{M})$
if $\mathcal{V}^* = -\infty$ **then**
    return "Infeasible"
else
    return π

- $\bar{\mathcal{A}}_h(s, \bar{c}) := \left\{a \in \mathcal{A} \mid \Pr_{c \sim C_h(s,a)}\left[\bar{c} + c \leq B\right] = 1\right\}$.
- $\bar{P}_h((s', \bar{c} + c) \mid (s, \bar{c}), a) := P_h(s' \mid s, a) C_h(c \mid s, a)$.
- $\overline{R}_h((s, \bar{c}), a) := R_h(s, a)$.
- $\bar{s}_0 := (s_0, 0)$.

Theorem 8 (Optimality)**.** *Algorithm 5 solves* (ANY) *and can be implemented to run in finite time.*

We see from Theorem 8 that an anytime-constrained cMDP can be solved using Algorithm 5. If M is known, the agent can directly construct $\bar{M}$ using Definition 7 and then solve $\bar{M}$ using any RL planning algorithm. If M is unknown, the agent can still solve $\bar{M}$ by replacing the call to $\pi_h(s, \bar{c})$ in Algorithm 6 by a call to any RL learning algorithm.

Corollary 2 (Reduction)**.** *An optimal policy for an anytime-constrained cMDP can be computed from Algorithm 5 paired with any RL planning or learning algorithm. Thus, anytime-constrained RL reduces to standard RL.*

Augmented Policies. Observe that any Markovian policy π for $\bar{M}$ is a *augmented policy* that maps (state, cost)-pairs to actions. This policy can be translated into a full history policy or can be used directly through the new interaction protocol

Algorithm 6 Augmented Interaction Protocol

Input: augmented policy π

$\bar{s}_1 = (s_0, 0)$ and $\bar{c}_1 = 0$.
for $h = 1$ to H **do**
  $a_h = \pi_h(\bar{s}_h)$.
  $c_h \sim C_h(s_h, a_h)$ and $s_{h+1} \sim P_h(s_h, a_h)$.
  $\bar{c}_{h+1} = \bar{c}_h + c_h$.
  $\bar{s}_{h+1} = (s_{h+1}, \bar{c}_{h+1})$.

described in Algorithm 6. By recording the cumulative cost, the agent effectively simulates the $\pi - \bar{M}$ interaction through the $\pi - M$ interaction.

Analysis. To understand augmented policies, we need new machinery than typical MDP theory. Since traditional policies are insufficient for anytime constraints, we need to directly compare against cost-history-dependent policies. However, we cannot consider arbitrary histories, since an infeasible history could allow higher value. Rather, we focus on histories that are induced by safe exploration:

$$W_h(s, \bar{c}) := \Big\{ \tau_h \in \mathcal{H}_h \mid \exists \pi,\ \mathbb{P}^{\pi}_{\tau_h}\left[s_h = s, \bar{c}_h = \bar{c}\right] = 1, \\ \mathbb{P}^{\pi}_{\tau_k}\left[\bar{c}_{k+1} \le B\right] = 1 \quad \forall k \in [h-1] \Big\}.$$

Here, $\mathbb{P}^{\pi}_{\tau_h}[\cdot] := \mathbb{P}^{\pi}_{M}[\cdot \mid \tau_h]$ and $\mathbb{E}^{\pi}_{\tau_h}[\cdot] := \mathbb{E}^{\pi}_{M}[\cdot \mid \tau_h]$ denote the conditional probability and expectation given partial history τ_h. The condition $\mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \le B] = 1$ enforces that any action taken along the trajectory never could have violated the budget.

We must also restrict to policies that are feasible given such a history: $\Pi_M(\tau_h) := \{\pi \mid \mathbb{P}^{\pi}_{\tau_h}\left[\forall k \in [H],\ \bar{c}_{k+1} \le B\right] = 1\}$. Note that generally $\Pi_M(\tau_h) \supset \Pi_M$ so some $\pi \in \Pi_M(\tau_h)$ need not be feasible, but importantly $\Pi_M(s_0) = \Pi_M$ contains only feasible policies. We define,

$$V^*_h(\tau_h) := \max_{\pi \in \Pi_M(\tau_h)} V^{\pi}_h(\tau_h)\ ; \mathcal{V}^*_h(s, \bar{c}) := \max_{\pi} \mathcal{V}^{\pi}_h(s, \bar{c}),$$

to be the optimal feasible value conditioned on τ_h, and the optimal value for $\bar{M}$ from time h onward starting from $(s, \bar{c})$, respectively. We show that optimal feasible solutions satisfy the *augmented bellman-optimality equations*.

Lemma 8. *For any* $h \in [H+1]$, $(s, \bar{c}) \in \bar{\mathcal{S}}_h$, *and* $\tau_h \in W_h(s, \bar{c})$, $\mathcal{V}_h^*(s, \bar{c}) = V_h^*(\tau_h)$.

The proof is more complex than the traditional bellman-optimality equations. It requires (1) backward induction to argue that the value is maximal under any safe partial history and (2) forward induction to argue the costs accrued respect the anytime constraints. It then follows that solutions to $\bar{M}$ are solutions to (ANY).

4.3.1 Complexity Analysis

To analyze the efficiency of our reduction, we define a combinatorial measure of a cMDP's complexity.

Definition 8 (Cost Diversity)**.** The *cost diversity*, $\mathcal{D}_M$, is the total number of distinct cumulative costs the agent could face at any time:

$$\mathcal{D}_M := \max_{h \in [H+1]} \left| \left\{ c \mid \exists s, (s, c) \in \bar{\mathcal{S}}_h \right\} \right|.$$

When clear from context, we refer to $\mathcal{D}_M$ as $\mathcal{D}$.

We call $\mathcal{D}$ the diversity as it measures the largest cost population that exists in any generation h. The diversity naturally captures the complexity of an instance since the agent would likely encounter at least this many cumulative costs when computing or learning an optimal policy using any safe approach.

In particular, we can bound the complexity of the planning and learning algorithms produced from our reduction in terms of the diversity. For concreteness, we pair

Algorithm 5 with backward induction [5] to produce a planning algorithm and with BPI-UCBVI [81] to produce a learning algorithm.

Proposition 10 (Complexity)**.** *Using Algorithm 5, an optimal policy for an anytime-constrained cMDP can be computed in* $O\left(HS^2An\mathcal{D}\right)$ *time and learned with* $\tilde{O}\left(H^3SA\mathcal{D}\log(\frac{1}{\delta})/\gamma^2\right)$ *sample complexity. Furthermore, the amount of space needed to store the policy is* $O\left(HS\mathcal{D}\right)$*.*

In the worst case, $\mathcal{D}$ could be exponentially large in the time horizon. However, for many cMDPs, $\mathcal{D}$ is small. One key factor in controlling the diversity is the precision needed to represent the supported costs in memory.

Lemma 9 (Precision)**.** *If the cost precision is at most* k*, then* $\mathcal{D} \leq H^d 2^{(k+1)d}$*.*

We immediately see that when the costs have precision k, all of our algorithms have complexity polynomial in the size of M and exponential in k and d. By definition, this means our algorithms are fixed-parameter tractable in k so long as d is held constant. Moreover, we see as long as the costs can be represented with logarithmic precision, our algorithms have polynomial complexity.

Theorem 9 (Fixed-Parameter Tractability)**.** *For constant* d*, if* $k = O(\log(|M|))$*, planning (learning) for anytime-constrained cMDPs can be performed in polynomial time (sample complexity) using Algorithm 5, and the computed policy can be stored with polynomial space.*

Remark 9. Many cost functions can be represented with small precision. In practice, all modern computers use fixed precision numbers. So, in any real system, our algorithms have polynomial complexity. Although technically efficient, our methods can be prohibitively expensive when k or d is large. However, this complexity seems unavoidable since computing exact solutions to (ANY) is NP-hard in general.

4.4 Approximation Algorithms

Since Theorem 7 rules out the possibility of traditional value-approximation algorithms due to the hardness of finding feasible policies, we relax the requirement of feasibility. We show that even with a slight relaxation of the constraint, solutions with optimal value can be found efficiently. Conversely, we can satisfy the constraint but with a weaker guarantee on value. Our approximate-feasibility methods can even handle infinite support distributions so long as they are bounded above.

Assumption 6. $c_{max} := \sup_{h,s,a} \sup C_h(s,a) < \infty$.

If $Hc_{max} \leq B$, then every policy is feasible, which just leads to a standard unconstrained problem. A similar phenomenon happens if $c_{max} \leq 0$. Thus, we assume WLOG that $Hc_{max} > B$ and $c_{max} > 0$.

Definition 9 (Approximate Feasibility)**.** For any $\epsilon > 0$, a policy π is ϵ-additive feasible if,

$$\mathbb{P}_M^\pi \left[\forall t \in [H],\ \sum_{h=1}^{t} c_h \leq B + \epsilon \right] = 1, \tag{4.1}$$

and ϵ-relative feasible if,

$$\mathbb{P}_M^\pi \left[\forall t \in [H],\ \sum_{h=1}^{t} c_h \leq B(1 + \epsilon\sigma_B) \right] = 1, \tag{4.2}$$

where σ_B is the sign of B[3].

Approximation. The key to reducing the complexity of our reduction is lowering the cost diversity. Rather than consider every cost that can be accumulated from safe exploration, the agent can consider a smaller set of approximate cumulative

[3]When the costs and budgets are negative, negating the constraint yields $\sum_{t=1}^{H} c_t \geq |B|\,(1-\epsilon)$, which is the traditional notion of relative approximation for covering objectives.

Algorithm 7 Approximate Reduction

Input: cMDP M and projection f

$\hat{M} \leftarrow$ Definition 10(M, f)
$\pi, \hat{V}^* \leftarrow \text{Solve}(\hat{M})$
if $\hat{V}^* = -\infty$ **then**
    return "Infeasible"
else
    return π

costs. Specifically, for any cumulative cost $\bar{c}_h$ and cost c_h, the agent considers some $\hat{c}_{h+1} = f_h(\bar{c}_h, c_h)$ instead of $\bar{c}_{h+1} = \bar{c}_h + c_h$.

We view f as projecting a cumulative cost onto a smaller approximate cost space. Following the principle of optimism, we also ensure that $f(\bar{c}_h, c_h) \leq \bar{c}_h + c_h$. This guarantees that any optimal policy under the approximate costs achieves optimal value at the price of a slight violation in the budget.

If the agent records the approximate costs induced by the projection f, the interaction evolves according to the following unconstrained MDP.

Definition 10 (Approximate MDP)**.** The *approximate MPD* $\hat{M} := (\hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{P}, \hat{R}, H, \hat{s}_0)$ where,

$$\hat{S}_{h+1} := \Big\{(s', \hat{c}') \mid \exists (s, \hat{c}) \in \hat{S}_h, a \in \mathcal{A}, c' \in C_h(s, a), \\ \hat{c}' = f_h(\hat{c}, c'), \Pr_{c \sim C_h(s,a)}[f_h(\hat{c}, c) \leq B] = 1, \\ P_h(s' \mid s, a) > 0\Big\},$$

is defined using approximate costs produced by safe exploration with a projection step. The other objects are defined analogously to Definition 7.

Our approximation algorithms, Algorithm 7, equate to solving $\hat{M}$ for different choices of f. To use any Markovian π for $\hat{M}$, the agent just needs to apply f when updating its approximate costs. In effect, the agent *accumulates then projects* to

create each approximate cost. The new interaction protocol is given by Algorithm 8.

To derive our choice of f, we first observe that the cumulative cost can never surpass B. Furthermore, should the agent ever accumulate a cost of $B - Hc_{max}$, it can no longer violate the budget along that trajectory. Thus, the agent's cumulative cost is always effectively within the interval $[B - Hc_{max}, B]$.

Projection. Our approach is to evenly subdivide the interval $[B - Hc_{max}, B]$ by length-ℓ intervals centered around 0. Then, the projection always maps a point in an interval to its left endpoint. Alternatively, we can think of ℓ as defining a new unit of measurement, and the projection maps each cumulative cost to its largest integer multiple of ℓ below the cumulative cost. Should the agent ever encounter an extremely negative cost, we safely truncate it to the projection of $B - (H - h)c_{max}$.

In symbols, we define our projection by, $f_h(\hat{c}, c) :=$

$$\begin{cases} \hat{c} + \left\lfloor \frac{c}{\ell} \right\rfloor \ell & \text{if } \hat{c} + c \geq B - (H-h)c_{max} \\ \left\lfloor \frac{B-(H-h)c_{max}}{\ell} \right\rfloor \ell & \text{o.w.} \end{cases}$$

Critically, the projection is defined so that each approximate cost is an underestimate of the true cost, but no farther than ϵ away from the true cost (except when a cost smaller than $B - (H - h)c_{max}$ is encountered).

Lemma 10. *For any feasible policy π for $\hat{M}$ and any $h \in [H+1]$, $\mathbb{P}^{\pi}_{M}[(\hat{c}_h \leq \bar{c}_h \leq \hat{c}_h + (h-1)\ell) \vee (\bar{c}_h, \hat{c}_h \leq B - (H-h+1)c_{max})] = 1$. Also, $\left|\left\{\hat{c}_h \mid \exists s \in \mathcal{S}, (s, \hat{c}_h) \in \hat{\mathcal{S}}_h\right\}\right| \leq \left(\frac{H\|c_{max}\|_\infty}{\ell} + 2\right)^d$.*

We see that solving $\hat{M}$ gives additive feasible solutions and $\hat{M}$ has far fewer states than $\bar{M}$.

Algorithm 8 Approximate Interaction Protocol

Input: policy π and projection f

$\hat{s}_1 = (s_0, 0)$ and $\hat{c}_1 = 0$.
for $h = 1$ to H **do**
  $a_h = \pi_h(\hat{s}_h)$
  $c_h \sim C_h(s_h, a_h)$ and $s_{h+1} \sim P_h(s_h, a_h)$
  $\hat{c}_{h+1} = f_h(\hat{c}_h, c_h)$
  $\hat{s}_{h+1} = (s_{h+1}, \hat{c}_{h+1})$

Theorem 10 (Approximation)**.** *Algorithm 7 computes an $H\ell$-additive feasible policy whose value is at least the optimal value of* (ANY) *and that can be stored with $O\left(H^{d+1} S \left\|c_{max}\right\|_\infty^d / \ell^d\right)$ space.*

Like with our original reduction, the interaction protocol in Algorithm 8 allows the agent to simulate $\hat{M}$ online through M. Thus, planning and learning in $\hat{M}$ can be done through M.

Remark 10. Note, the agent does not need to construct $\hat{S}$ using Definition 10; it suffices to consider the finite, stationary state space $\mathcal{S} \times \mathcal{C}$, where $\mathcal{C}$ is the ℓ-cover of $[B - Hc_{max}, B]$ defined by f. Technically, for learning, the agent should already know or have learned a bound on c_{max} to know the approximate state space.

4.4.1 Approximation Guarantees

We can use Algorithm 7 with different choices of ℓ to achieve the traditional approximation guarantees defined in Definition 9.

Additive Approximation. Given any $\epsilon > 0$, we can compute an ϵ-additive feasible solution by choosing $\ell := \frac{\epsilon}{H}$. This approach is efficient so long as c_{max} is not too large, since c_{max} controls the number of discretized costs we need to consider.

Corollary 3 (Additive Reduction)**.** *For any $\epsilon > 0$, an optimal value, ϵ-additive feasible policy for an anytime-constrained cMDP can be computed in $O\big(H^{4d+1} S^2 A \left\|c_{max}\right\|_\infty^{2d}$*

$/\epsilon^{2d}\big)$ time and learned with $\tilde{O}\big(H^{2d+3}SA\,\|c_{max}\|_\infty^d \log(\frac{1}{\delta})/\gamma^2\epsilon^d\big)$ sample complexity using Algorithm 7 with $\ell := \frac{\epsilon}{H}$. Furthermore, the amount of space needed to store the policy is $O\left(H^{2d+1}S\,\|c_{max}\|_\infty^d/\epsilon^d\right)$. Thus, if d is constant and $c_{max} \leq \text{poly}(|M|)$, our additive methods are polynomial time and sample complexity.

Relative Approximation. Given any $\epsilon > 0$, we can compute an ϵ-relative feasible solution by choosing $\ell := \frac{\epsilon|B|}{H}$. This approach is efficient so long as c_{max} is not much larger than $|B|$. This allows us to capture cost ranges that are polynomial multiples of $|B|$, which could be exponentially large, unlike the additive approximation which requires that c_{max} to be polynomial.

Corollary 4 (Relative Reduction)**.** *For any $\epsilon > 0$, if $c_{max} \leq x\,|B|$, an optimal value, ϵ-relative feasible policy for an anytime-constrained cMDP can be computed in $O\left(x^{2d}H^{4d+1}S^2A/\epsilon^{2d}\right)$ time and learned with $\tilde{O}\left(x^dH^{2d+3}SA/\epsilon^d\log(\frac{1}{\delta})/\gamma^2\right)$ sample complexity using Algorithm 7 with $\ell = \frac{\epsilon|B|}{H}$. Furthermore, the amount of space needed to store the policy is $O\left(x^dH^{2d+1}S/\epsilon^d\right)$. Thus, if d is constant and $x \leq \text{poly}(|M|)$, our methods are polynomial time and sample complexity.*

Corollary 5. *If all costs are positive, the H dependence in each guarantee of both the additive and relative approximation improves to H^{2d+1}, H^{d+3}, and H^{d+1}, respectively.*

Limitations. Using our additive approximation, we can efficiently handle any cMDP with $c_{max} \leq \text{poly}(|M|)$. Using the relative approximation, we can even handle the case that c_{max} is exponentially large so long as $c_{max} \leq \text{poly}(|M|)\,|B|$. Thus, we can efficiently compute approximately feasible solutions so long as $c_{max} \leq \text{poly}(|M|)\max(1, |B|)$.

We point out that the condition $c_{max} \leq \text{poly}(|M|)\,|B|$ is very natural. If the costs all have the same sign, any feasible policy induces costs with $c_{max} \leq |B|$. In

our driving example, the condition simply says the vehicle cannot use more fuel than the capacity of the tank. In fact, this bottleneck is not due to our approach; some bound on c_{max} is necessary for efficient computation as Proposition 11 shows.

Proposition 11. *For any fixed $\epsilon > 0$, computing an optimal-value, ϵ-additive or ϵ-relative feasible solution to the knapsack problem with negative weights is NP-hard. Hence, it is hard for anytime-constrained cMDPs.*

Feasibility Scheme. Let $OPT(B)$ denote the optimal value V^* of an anytime-constrained cMDP with budget B. Algorithm 7 provides an efficient approximation and guarantees at least $OPT(B)$ value but with the possibility of slightly going over budget, up to $B + \epsilon$ or $B(1 + \epsilon)$. If it is important that the budget B is never violated, we can use the same approximate algorithm with one change: We instead give it $\hat{M}'$, which is $\hat{M}$ constructed from M under a smaller budget: (1) $B - \epsilon$ for the additive approximation and (2) $B/(1 + \epsilon)$ for the relative approximation. Then, any over-budget by Algorithm 7 is compensated by the smaller budget, so that the cumulative cost is still under B. Thus we have both efficiency and (budget B) feasibility. The drawback is that the algorithm now only guarantees a value at least $OPT(B - \epsilon)$ or $OPT(B/(1 + \epsilon))$, both can be much smaller than $OPT(B)$.

Proposition 12 (Feasible Solutions)**.** *If π is returned by Algorithm 7 using $\ell = \frac{\epsilon}{H}$ ($\ell = \frac{\epsilon|B|}{H}$) with budget $B' = B - \epsilon$ ($B' = \frac{B}{1+\epsilon}$), then π is feasible for* (ANY).

4.5 Conclusion

In this chapter, we formalized and rigorously studied anytime-constrained cMDPs. Although traditional policies cannot solve anytime-constrained cMDPs, we showed that deterministic augmented policies suffice. We also presented a fixed-parameter

tractable reduction based on cost augmentation and safe exploration that yields efficient planning and learning algorithms when the cost precision is $O(\log(|M|))$. In addition, we developed efficient planning and learning algorithms to find ϵ-approximately feasible policies with optimal value whenever the maximum supported cost is $O(\text{poly}(|M|)\max(1, |B|))$. Based on our hardness of approximation results, this is the best approximation guarantee we can hope for under worst-case analysis.

Chapter 5

FPTAS for One Constraint

Abstract. In this chapter, we present a novel algorithm that efficiently computes near-optimal deterministic policies for constrained reinforcement learning (CRL) problems. Our approach combines three key ideas: (1) value-demand augmentation, (2) action-space approximate dynamic programming, and (3) time-space rounding. Our algorithm constitutes a fully polynomial-time approximation scheme (FPTAS) for any time-space recursive (TSR) cost criteria. A TSR criteria requires the cost of a policy to be computable recursively over both time and (state) space, which includes classical expectation, almost sure, and anytime constraints. Our work answers three open questions spanning two long-standing lines of research: polynomial-time approximability is possible for 1) anytime-constrained policies, 2) almost-sure-constrained policies, and 3) deterministic expectation-constrained policies.

5.1 Introduction

Constrained Reinforcement Learning (CRL) traditionally produces stochastic, expectation constrained policies that can behave undesirably - imagine a self-driving car

that randomly changes lanes or runs out of fuel. However, artificial decision-making systems must be predictable, trustworthy, and robust. One approach to ensuring these qualities is to focus on deterministic policies, which are inherently predictable, easily implemented [36], reliable for autonomous vehicles [56, 40], and effective for multi-agent coordination [88]. Similarly, almost sure and anytime constraints [79] provide inherent trustworthiness and robustness, essential for applications in medicine [26, 87, 64], disaster relief [35, 117, 109], and resource management [75, 70, 90, 13]. Despite the advantages of deterministic policies and stricter constraints, even the computation of approximate solutions has remained an open challenge since NP-hardness was proven nearly 25 years ago [36]. Our work addresses this challenge by studying the computational complexity of computing deterministic policies for general constraint criteria.

Consider a constrained Markov Decision Process (cMDP) denoted by M. Let C represent an arbitrary cost criterion and B be the available budget. We focus on the set of deterministic policies denoted by Π^D. Our objective is to compute: $\max_{\pi \in \Pi^D} V_M^\pi$ s.t. $C_M^\pi \leq B$, where V_M^π is the value and C_M^π is the cost of π in M. This objective generalizes the example of a self-driving car calculating the fastest fixed route without running out of fuel. Our main question is the following:

> *Can near-optimal deterministic policies for constrained reinforcement learning problems be computed in polynomial time?*

Although optimal stochastic policies for expectation-constrained problems are efficiently computable [5], the situation drastically changes when we require deterministic policies and general constraints. Computing optimal deterministic policies is NP-hard for most popular constraints, including expectation [36], chance [123], almost sure, and anytime constraints [79]. This complexity remains even if we relax our goal to finding just one feasible policy, provided that we are dealing with a single

chance constraint [123], or at least two of the other mentioned constraints [79]. Beyond these computational challenges, traditional solution methods, such as backward induction [91, 5], fail to apply due to the cyclic dependencies among subproblems: the value of any decision may depend on the costs of both preceding and concurrent decisions, preventing a solution from being computed in a single backward pass.

Past Work. Past approaches fail to simultaneously achieve computational efficiency, feasibility, and optimality. Optimal and feasible algorithms, albeit inefficient, utilize Mixed-Integer Linear Programs [32] and Dual-guided heuristic forward searches [55] for expectation-constraints, and cost-augmented MDPs for almost sure [18] and anytime constraints [79]. Conversely, optimal and efficient, though infeasible, algorithms are known for expectation [105], almost sure, and anytime constraints [79]. A fully polynomial-time approximation scheme (FPTAS) [116] is known for expectation constraints, but it requires strong assumptions such as a constant horizon [63]. Balancing computational efficiency, feasibility, and optimality remains the bottleneck to efficient approximation.

Our Contributions. We present an FPTAS for computing deterministic policies under any time-space recursive (TSR) constraint criteria. A TSR criteria requires the cost of a policy to be computable recursively in both time and (state) space, which captures expectation, almost sure, and anytime constraints. Since non-TSR criteria, such as chance constraints [123], are provably inapproximable, TSR seems pivotal for efficient computation. Overall, our general framework answers three open complexity questions spanning two longstanding lines of work: we prove polynomial-time approximability for 1) anytime-constrained policies, 2) almost-sure-constrained policies, and 3) deterministic expectation-constrained policies, which have been open for nearly 25 years [36].

Our approach breaks down into three main ideas: (1) value-demand augmentation, (2) action-space approximate dynamic programming, and (3) time-space rounding. We augment the states with value demands and the actions with future value demands to break cyclic subproblem dependencies, enabling dynamic programming methods. Importantly, we use values because they can be rounded without compromising feasibility [79] and can capture constraints that are not predictable from cumulative costs. However, this results in an exponential action space that makes solving the Bellman operator as hard as the knapsack problem. By exploiting the space-recursive nature of the criterion, we can efficiently approximate the Bellman operator with dynamic programming. Finally, rounding value demands result in approximation errors over both time and space, but carefully controlling these errors ensures provable guarantees.

5.1.1 Related Work

Approximate Packing. Many stochastic packing problems, which generalize the knapsack problem, are captured by our problem. Dean et al. [30], Frieze and Clarke [38] derived optimal approximation ratio algorithms for stochastic packing and integer packing with multiple constraints, respectively. Yang et al. [125], Bhalgat et al. [12] designed efficient approximation algorithms for variations of the stochastic knapsack problem. Then, Halman et al. [48] derived an FPTAS for a general class of stochastic dynamic programs, which was then further improved in [47, 3]. These methods require a single-dimensional state space that captures the constraint. In contrast, our problems have an innate state space in addition to the constraint. Our work forms a similar general dynamic programming framework for the more complex MDP setting.

Constrained RL. It is known that stochastic expectation-constrained policies are polynomial-time computable via linear programming [5], and many planning and learning algorithms exist for them [89, 110, 14, 53]. Some learning algorithms can even avoid violation during the learning process under certain assumptions [115, 7]. Furthermore, Brantley et al. [17] developed no-regret algorithms for cMDPs and extended their algorithms to the setting with a constraint on the cost accumulated over all episodes, which is called a knapsack constraint [17, 22].

Safe RL. The safe RL community [39, 45] has mainly focused on no-violation learning for stochastic expectation-constrained policies [24, 16, 4, 21, 10] and solving chance constraints [114, 132], which capture the probability of entering unsafe states. Performing learning while avoiding dangerous states [132] is a special case of expectation constraints that has also been studied [96, 107] under non-trivial assumptions. In addition, instantaneous constraints, which require the expected cost to be within budget at all times, have also been studied [69, 37, 43].

5.2 Cost Criteria

In this section, we formalize our problem setting. We also define our conditions for cost criteria.

Constrained Markov Decision Processes. A (tabular, finite-horizon) *Constrained Markov Decision Process* (cMDP) is a tuple $M = (\mathcal{S}, \mathcal{A}, P, r, c, H)$, where (i) $\mathcal{S}$ is the finite set of *states*, (ii) $\mathcal{A}$ is the finite set of *actions*, (iii) $P_h(s, a) \in \Delta(S)$ is the *transition* distribution, (iv) $r_h(s, a) \in \mathbb{R}$ is the *reward*, (v) $c_h(s, a) \in \mathbb{R}$ is the *cost*, and (vi) $H \in \mathbb{N}$ is the finite *time horizon*. We let $S := |\mathcal{S}|$, $A := |\mathcal{A}|$, $[H] := \{1, \ldots, H\}$, and $\mathcal{M}$ denote the set of all cMDPs. We also let $r_{max} \stackrel{\text{def}}{=} \max_{h,s,a} |r_h(s, a)|$ denote

the maximum magnitude reward, $r_{min} \stackrel{\text{def}}{=} \min_{h,s,a} r_h(s,a)$ denote the true minimum reward, and $p_{min} \stackrel{\text{def}}{=} \min_{h,s,a,s'} P_h(s' \mid s,a)$ denote the minimum transition probability. Since $\mathcal{S}$ is a finite set, we often assume $\mathcal{S} = [S]$ WLOG. Lastly, for any predicate p, we use the Iverson bracket notation $[p]$ to denote 1 if p is true and 0 otherwise, and we let χ_p denote the characteristic function which evaluates to 0 if p is true and ∞ otherwise.

Interaction Protocol. The agent interacts with M using a policy $\pi = (\pi_h)_{h=1}^H$. In the fullest generality, $\pi_h : \mathcal{H}_h \to \Delta(\mathcal{A})$ is a mapping from the observed history at time h to a distribution of actions. In contrast, a deterministic policy takes the form $\pi_h : \mathcal{H}_h \to \mathcal{A}$. We let Π denote the set of all possible policies and Π^D denote the set of all deterministic policies. The agent starts at the initial state $s_0 \in \mathcal{S}$ with observed history $\tau_1 = (s_0)$. For any $h \in [H]$, the agent chooses an action $a_h \sim \pi_h(\tau_h)$. Then, the agent receives immediate reward $r_h(s_h, a_h)$ and cost $c_h(s_h, a_h)$. Lastly, M transitions to state $s_{h+1} \sim P_h(s_h, a_h)$ and the agent updates the history to $\tau_{h+1} = (\tau_h, a_h, s_{h+1})$. This process is repeated for H steps; the interaction ends once s_{H+1} is reached.

Objective. For any cost criterion $C : \mathcal{M} \times \Pi \to \mathbb{R}$ and budget $B \in \mathbb{R}$, the agent's goal is to compute a solution to the following optimization problem:

$$\max_{\pi \in \Pi} \mathbb{E}_M^\pi \left[\sum_{h=1}^H r_h(s_h, a_h) \right] \quad \text{s.t.} \quad \begin{cases} C_M^\pi \le B \\ \pi \text{ deterministic} \end{cases} . \tag{CON}$$

Here, $\mathbb{P}_M^\pi$ denotes the probability law over histories induced from the interaction of π with M, and $\mathbb{E}_M^\pi$ denotes the expectation defined by this law. We let $V_M^\pi \stackrel{\text{def}}{=} \mathbb{E}_M^\pi \left[\sum_{t=1}^H r_t(s_t, a_t) \right]$ denote the value of a policy π, and V_M^* denote the optimal

solution value to (CON).

Cost criteria. We consider a broad family of cost criteria that satisfy a strengthening of the standard policy evaluation equations [91]. This strengthening requires not only the cost of a policy to be computable recursively in the time horizon, but at each time the cost should also break down recursively in (state) space.

Definition 11 (TSR)**.** We call a cost criterion C *time-recursive* (TR) if for any cMDP M and policy $\pi \in \Pi^D$, π's cost decomposes recursively into $C_M^\pi = C_1^\pi(s_0)$. Here, $C_{H+1}^\pi(\cdot) = \mathbf{0}$ and for any $h \in [H]$ and $\tau_h \in \mathcal{H}_h$,

$$C_h^\pi(\tau_h) = c_h(s,a) + f\left(\left(P_h(s' \mid s,a), C_{h+1}^\pi(\tau_h, a, s')\right)_{s' \in P_h(s,a)}\right), \tag{TR}$$

where $s = s_h(\tau_h)$, $a = \pi_h(\tau_h)$, and f is a non-decreasing function[1] computable in $O(S)$ time. For technical reasons, we also require that $f(x) = \infty$ whenever $\infty \in x$.

We further say C is *time-space-recursive* (TSR) if the f term above is equal to $g_h^{\tau_h,a}(1)$. Here, $g_h^{\tau_h,a}(S+1) = 0$ and for any $t \le S$,

$$g_h^{\tau_h,a}(t) = \alpha\left(\beta\left(P_h(t \mid s,a), C_{h+1}^\pi(\tau_h,a,t)\right), g_h^{\tau_h,a}(t+1)\right), \tag{SR}$$

where α is a non-decreasing function, and both α, β are computable in $O(1)$ time. We also assume that $\alpha(\cdot,\infty) = \infty$, and β satisfies $\alpha(\beta(0,\cdot),x) = x$ to match f's condition.

Since the TR condition is a slight generalization of traditional policy evaluation, it is easy to see that we can solve for minimum-cost policies using backward induction.

[1] When we say a multivariate function is non-decreasing, we mean it is non-decreasing with respect to the partial ordering induced by component-wise ordering.

Proposition 13 (TR Intuition)**.** *If C is TR, then C satisfies the usual optimality equations. Furthermore,* $\arg\min_{\pi\in\Pi^D} C_M^\pi$ *can be computed using backward induction in $O(HS^2A)$ time.*

Although the TR condition is straightforward, the TSR condition is more strict. We will see the utility of the TSR condition in Section 5.4 when computing Bellman updates. For now, we point out that the TSR condition is not too restrictive: it is satisfied by many popular criteria studied in the literature.

Proposition 14 (TSR examples)**.** *The following classical constraints can be modeled by a TSR cost constraint.*

1. (Expectation Constraints) *are captured by* $C_M^\pi \overset{def}{=} \mathbb{E}_M^\pi\left[\sum_{h=1}^H c_h(s_h,a_h)\right] \le B$. *We see C is TSR by defining* $\alpha(x,y) \overset{def}{=} x+y$ *and* $\beta(x,y) \overset{def}{=} xy$.

2. (Almost Sure Constraints) *are captured by* $C_M^\pi \overset{def}{=} \max_{\tau\in\mathcal{H}_{H+1},\, \mathbb{P}_M^\pi[\tau]>0} \sum_{h=1}^H c_h(s_h,a_h) \le B$. *We see C is TSR by defining* $\alpha(x,y) \overset{def}{=} \max(x,y)$ *and* $\beta(x,y) \overset{def}{=} [x>0]y$.

3. (Anytime Constraints) *are captured by* $C_M^\pi \overset{def}{=} \max_t \max_{\tau\in\mathcal{H}_{H+1},\, \mathbb{P}_M^\pi[\tau]>0} \sum_{h=1}^t c_h(s_h,a_h) \le B$. *We see C is TSR by defining* $\alpha(x,y) \overset{def}{=} \max(0,\max(x,y))$ *and* $\beta(x,y) \overset{def}{=} [x>0]y$.

Remark 11 (Extensions). Our methods can also handle stochastic costs and infinite discounting. We defer the details to Appendix D.5. Moreover, we can handle multiple constraints using vector-valued criteria so long as the comparison operator is a total ordering of the vector space.

Remark 12 (Inapproximability). Our methods cannot handle chance constraints or more than one of our example constraints. However, this is not a limitation of our framework as the problem becomes provably inapproximable under said constraints [123, 79].

5.3 Covering Algorithm

In this section, we propose an algorithm to solve (CON). Our approach relies on converting the original problem into an equivalent covering problem that can be solved using an unconstrained MDP. This covering MDP is derived using the key idea of value augmentation.

Packing and Covering. We can view (CON) as a *packing program*, which wishes to maximize V_M^π subject to $C_M^\pi \leq B$. However, we could also tackle the problem by reversing the objective: attempt to minimize C_M^π subject to $V_M^\pi \geq V_M^*$. If (CON) is feasible, then any optimal solution π to this *covering program* satisfies $V_M^\pi \geq V_M^*$ and $C_M^\pi \leq B$. Thus, we can solve the original packing program by solving the covering program.

Proposition 15 (Packing-Covering Reduction)**.** *Suppose that* $C_M^* \stackrel{def}{=} \min_{\pi \in \Pi^D} C_M^\pi$ *s.t.* $V_M^\pi \geq V_M^*$. *Then,* $C_M^* \leq B \iff V_M^* > -\infty$. *Furthermore, if* $V_M^* > -\infty$, *then,*

$$\begin{array}{c} \arg\min_{\pi \in \Pi^D} C_M^\pi \\ V_M^\pi \geq V_M^* \end{array} \subseteq \begin{array}{c} \arg\max_{\pi \in \Pi^D} V_M^\pi \\ C_M^\pi \leq B \end{array}. \tag{PC}$$

Thus, any solution to the covering program is a solution to the packing program.

We focus on the covering program for several reasons. To optimize the value recursively, we would need to predict the final cost resulting from intermediate decisions to ensure feasibility. Generally, such predictions would require strict assumptions on the cost criteria. By treating the value as the constraint instead, we only need to assume the cost can be optimized efficiently. Moreover, values are well understood in RL and are more amenable to approximation [79]. Thus, the covering program allows us to capture many criteria, ensure feasibility, and compute accurate

value approximations.

Value Augmentation. We can solve the covering program by solving a cost-minimizing MDP $\bar{M}$. The key idea is to augment the state space with value demands, (s, v). Then, the agent can recursively reason how to minimize its cost while meeting the current value demand. If the agent starts at (s_0, V^*_M), then an optimal policy for $\bar{M}$ should be a solution to the covering program.

The key invariant we desire is that any feasible policy π for $\bar{M}$ should satisfy $\bar{V}^\pi_h(s, v) \geq v$. To ensure this invariance, we recall the policy evaluation equations [91]. If $\pi_h(s) = a$, then,

$$\bar{V}^\pi_h(s, v) = r_h(s, a) + \sum_{s'} P_h(s' \mid s, a)\bar{V}^\pi_{h+1}(s', v_{s'}). \tag{PE}$$

For the value invariant to be satisfied, it suffices for the agent to choose an action a and commit to future value demands $v_{s'}$ satisfying,

$$r_h(s, a) + \sum_{s'} P_h(s' \mid s, a)v_{s'} \geq v. \tag{DEM}$$

We can view choosing future value demands as part of the agent's augmented actions. Then, at any augmented state (s, v), the agent's augmented action space includes all $(a, \mathbf{v}) \in \mathcal{A} \times \mathbb{R}^S$ satisfying (DEM). When M transitions to $s' \sim P_h(s, a)$, the agent's new augmented state should consist of the environment's new state in addition to its chosen demand for that state, $(s', v_{s'})$. Putting these pieces together yields the definition of the cover MDP, Definition 12.

Definition 12 (Cover MDP)**.** The *cover MDP* $\bar{M} \stackrel{\text{def}}{=} (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{P}, \bar{c}, H)$ where,

1. $\bar{\mathcal{S}} \stackrel{\text{def}}{=} \mathcal{S} \times \mathcal{V}$ where $\mathcal{V} \stackrel{\text{def}}{=} \left\{v \mid \exists \pi \in \Pi^D, h \in [H+1], \tau_h \in \mathcal{H}_h,\ V^\pi_h(\tau_h) = v\right\}$

Algorithm 9 Reduction to RL

Input: (M, C, B)

$\bar{M}, \bar{C} \leftarrow$ Definition 12(M, C)
$\pi, \bar{C}^* \leftarrow \textsc{Solve}(\bar{M}, \bar{C})$
if $\bar{C}_1^*(s_0, v) > B$ for all $v \in \mathcal{V}$ **then**
    return "Infeasible"
else
    return π

Algorithm 10 Augmented Interaction

Input: π

$\bar{s}_1 = (s_0, V_M^*)$
for $h \leftarrow 1$ to H **do**
    $(a, \mathbf{v}) \leftarrow \pi_h(\bar{s}_h)$
    $r_h = r_h(s, a)$ and $s_{h+1} \sim P_h(s_h, a)$
    $\bar{s}_{h+1} = (s_{h+1}, v_{s_{h+1}})$

2. $\bar{\mathcal{A}}_h(s, v) \stackrel{\text{def}}{=} \left\{(a, \mathbf{v}) \in \mathcal{A} \times \mathcal{V}^S \mid r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) v_{s'} \geq v\right\}$.

3. $\bar{P}_h((s', v') \mid (s, v), (a, \mathbf{v})) \stackrel{\text{def}}{=} P_h(s' \mid s, a)[v' = v_{s'}]$.

4. $\bar{c}_h((s, v), (a, \mathbf{v})) \stackrel{\text{def}}{=} c_h(s, a)$.

The objective for $\bar{M}$ is to minimize the cost function $\bar{C} \stackrel{\text{def}}{=} C_{\bar{M}}$ with modified base case $\bar{C}_{H+1}^{\pi}(s, v) \stackrel{\text{def}}{=} \chi_{\{v \leq 0\}}$.

Covering Algorithm. Importantly, the action space definition ensures the value constraint is satisfied. Meanwhile, the minimum cost objective ensures optimal cost. So long as our cost is TR, $\bar{M}$ can be solved using fast RL methods instead of the brute force computation required for general covering programs. These properties ensure our method, Algorithm 9, is correct.

Theorem 11 (Reduction)**.** *If* Solve *is any finite-time MDP solver, then Algorithm 9 correctly solves* (CON) *in finite time for any TR cost criterion.*

Remark 13 (Execution). Given a value-augmented policy π output from Algorithm 9, the agent can execute π using Algorithm 10. To compute V_M^* as the starting value, it suffices for the agent to compute,

$$V_M^* = \max \left\{ v \in \mathcal{V} \mid \bar{C}_1^*(s_0, v) \leq B \right\}. \tag{5.1}$$

This computation can be easily done given $\bar{C}_1^*(s_0, \cdot)$ in $O(|\mathcal{V}|)$ time.

5.4 Fast Bellman Updates

In this section, we present an algorithm to solve $\bar{M}$ from Definition 12 efficiently. Although the Bellman updates can be as hard to solve as the knapsack problem, we use ideas from knapsack approximation algorithms to create an efficient method. Our approach exploits (SR) through approximate dynamic programming on the action space.

Even if $\mathcal{V}$ were small, solving $\bar{M}$ would still be challenging due to the exponentially large action space. Even a single Bellman update requires the solution of a constrained optimization problem:

$$\begin{aligned} \bar{C}_h^*(s, v) = \min_{a, \mathbf{v}} \; & c_h(s, a) + f\left(\left(P_h(s' \mid s, a), \bar{C}_{h+1}^*(s', v_{s'}) \right)_{s' \in P_h(s,a)} \right) \\ \text{s.t. } & r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) v_{s'} \geq v. \end{aligned} \tag{BU}$$

Above, we used the fact that $(s', v') \in \bar{P}_h((s, v), (a, \mathbf{v}))$ iff $s' \in P_h(s, a)$ and $v' = v_{s'}$ to simplify f's input. Observe that even when each $v_{s'}$ only takes on two possible values, $\{0, w_{s'}\}$, the optimization above can capture the minimization version of the knapsack problem, implying that it is NP-hard to compute.

Recursive Approach. Fortunately, we can use the connection to the Knapsack problem positively to efficiently approximate the Bellman update. For any fixed $(s, v) \in \bar{\mathcal{S}}$ and $a \in \mathcal{A}$, we focus on the inner constrained minimization over $\mathbf{v}$:

$$\min_{\substack{\mathbf{v} \in \mathcal{V}^S, \\ r_h(s,a) + \sum_{s'} P_h(s' \mid s,a) v_{s'} \geq v}} f\left(\left(P_h(s' \mid s,a), \bar{C}^*_{h+1}(s', v_{s'})\right)_{s' \in P_h(s,a)}\right) \tag{5.2}$$

We use (SR) to transform this minimization over $\mathbf{v}$ into a sequential decision-making problem that decides each $v_{s'}$. As above, we can use the definition of $\bar{P}$ to simplify $g_h^{(s,v),(a,\mathbf{v})}(t, v')$ into a function of t alone:

$$g_h^{(s,v),(a,\mathbf{v})}(t) = \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t, v_t)\right), g_h^{(s,v),(a,\mathbf{v})}(t+1)\right). \tag{5.3}$$

Since v only constrains the valid $(a, \mathbf{v})$ pairs, we can discard v and use the simplified notation $g^{s,a}_{h,\mathbf{v}}(t)$ instead of $g_h^{(s,v),(a,\mathbf{v})}(t)$. It is then clear that we can recursively optimize the value of v_t by focusing on $g^{s,a}_{h,\mathbf{v}}(t)$.

To recursively encode the value constraint, we can record the partial value $u = r_h(s,a) + \sum_{s'=1}^{t-1} P_h(s' \mid s,a) v_{s'}$ that we have accumulated so far. Then, we can check if our choices for $\mathbf{v}$ satisfied the constraint with the inequality $u \geq v$. The formal recursion is defined in Definition 13.

Definition 13. For any $h \in [H]$, $s \in \mathcal{S}$, $v \in \mathcal{V}$, and $u \in \mathbb{R}$, we define, $g^{s,a}_{h,v}(S+1, u) = \chi_{\{u \geq v\}}$ and for $t \leq S$,

$$g^{s,a}_{h,v}(t, u) = \min_{v_t \in \mathcal{V}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t, v_t)\right), g^{s,a}_{h,v}(t+1, u + P_h(t \mid s,a) v_t)\right). \tag{DP}$$

Recursive Rounding. This approach can still be slow due to the exponential number of partial values u induced. Similarly to the knapsack problem, the key is to round each input u to ensure fewer subproblems. Unlike the knapsack problem,

however, we do not have an easily computable lower bound on the optimal value. Thus, we turn to a more aggressive recursive rounding. Since rounding may cause originally feasible values to violate the demand constraint, we also relax the demand constraint to $u \geq \kappa(v)$ for some lower bound function κ.

Definition 14. Fix a rounding function $\lfloor \cdot \rfloor_{\mathcal{G}}$ and a lower bound function κ. For any $h \in [H]$, $s \in \mathcal{S}$, $v \in \mathcal{V}$, and $u \in \mathbb{R}$, we define, $\hat{g}^{s,a}_{h,v}(S+1, u) = \chi_{\{u \geq v\}}$ and for $t \leq S$,

$$\hat{g}^{s,a}_{h,v}(t,u) \stackrel{\text{def}}{=} \min_{v_t \in \mathcal{V}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t, v_t)\right), \hat{g}^{s,a}_{h,v}(t+1, \lfloor u + P_h(t \mid s,a) v_t \rfloor_{\mathcal{G}}\right). \tag{ADP}$$

Fortunately, the approximate version behaves similarly to the original. The main difference is the constraint now ensures the rounded sums are at least the value lower bound. This is formalized in Lemma 11.

Lemma 11. *For any $t \in [S+1]$ and $u \in \mathbb{R}$, we have that,*

$$\begin{aligned} \hat{g}^{s,a}_{h,v}(t,u) = \min_{\mathbf{v} \in \mathcal{V}^{S-t+1}} \; & g^{s,a}_{h,\hat{\mathbf{v}}}(t) \\ s.t. \quad & \hat{\sigma}^{s,a}_{h,\mathbf{v}}(t,u) \geq \kappa(v), \end{aligned} \tag{5.4}$$

where $\hat{\sigma}^{s,a}_{h,\mathbf{v}}(t,u) \stackrel{def}{=} \left\lfloor \lfloor u + P_h(t \mid s,a) v_t \rfloor_{\mathcal{G}} + \ldots + P_h(S \mid s,a) v_S \right\rfloor_{\mathcal{G}}$.

To turn this recursion into a usable dynamic programming algorithm, we must also pre-compute the inputs to any sub-computation. Unlike in standard RL, this computation must be done with a forward recursion. The details for the approximate Bellman update are given in Definition 15.

Definition 15 (Approx Bellman)**.** For any $h \in [H]$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$, we define

Algorithm 11 Approx Bellman Update

Input: $(h, s, v, \bar{C}^*_{h+1})$

for $a \in \mathcal{A}$ **do**
  $\hat{g}^{s,a}_{h,v}(S+1, u) \leftarrow \chi_{\{u \geq v\}}\ \forall u \in \hat{\mathcal{U}}^{s,a}_h(S+1)$
  for $t \leftarrow S$ down to 1 **do**
    for $u \in \hat{\mathcal{U}}^{s,a}_h(t)$ **do**
      $v_{t,a}, \hat{g}^{s,a}_{h,v}(t, u) \leftarrow$ (ADP)
$a^*, \hat{C}^*_h(s, v) \leftarrow \min_{a \in \mathcal{A}} c_h(s, a) + \hat{g}^{s,a}_{h,v}(1, r_h(s, a))$
return $(a^*, v_{1,a^*}, \ldots, v_{S,a^*})$ and $\hat{C}^*_h(s, v)$

Algorithm 12 Approx Solve

Input: $(\bar{M}, \bar{C})$

$\hat{C}^*_{H+1}(s, v) \leftarrow \chi_{\{v \leq 0\}}$ for all $(s, v) \in \bar{\mathcal{S}}$
for $h \leftarrow H$ down to 1 **do**
  for $(s, v) \in \bar{\mathcal{S}}$ **do**
    $\hat{a}, \hat{C}^*_h(s, v) \leftarrow$ Algorithm 11$(h, s, v, \hat{C}^*_{h+1})$
    $\pi_h(s, v) \leftarrow \hat{a}$
return π and $\hat{C}^*$

$\hat{\mathcal{U}}^{s,a}_h(1) \stackrel{\text{def}}{=} \{r_h(s, a)\}$ and for any $t \in [S]$,

$$\hat{\mathcal{U}}^{s,a}_h(t+1) \stackrel{\text{def}}{=} \bigcup_{v_t \in \mathcal{V}} \bigcup_{u \in \hat{\mathcal{U}}^{s,a}_h(t)} \left\{ \lfloor u + P_h(t \mid s, a) v_t \rfloor_{\mathcal{G}} \right\}. \tag{5.5}$$

Then, an approximation to the Bellman update can be computed using Algorithm 11.[2]

Proposition 16. *Algorithm 12 runs in $O(HS^2A|\mathcal{V}|^2\hat{U})$ time, where $\hat{U} \stackrel{def}{=} \max_{h,s,a} |\hat{U}^{s,a}_h|$. When $\lfloor \cdot \rfloor_{\mathcal{G}}$ and κ are the identity function, Algorithm 12 outputs an optimal solution to $\bar{M}$.*

Remark 14 (Speedups). The runtime of our methods can be quadratically improved by rounding the differences instead of the sums. We defer the details to Appendix D.5.

[2]We use the notation $x, o \leftarrow \min_x z(x)$ to say that x is the minimizer and o the value of the optimization.

5.5 Approximation Algorithms

In this section, we present our approximation algorithms for solving (CON). We carefully round the value demands over both time and space to induce an approximate MDP. Solving this approximate MDP with Algorithm 12 yields our FPTAS.

Although we can avoid exponential-time Bellman updates, the running time of the approximate Bellman update will still be slow if $|\mathcal{V}|$ is large. To reduce the complexity, we instead use a smaller set of approximate values by rounding elements of $|\mathcal{V}|$. By rounding down, we effectively relax the value-demand constraint. More aggressive rounding not only leads to smaller augmented state spaces but also to smaller cost policies. The trade-off is aggressive rounding leads to weaker guarantees on the computed policy's value. Thus, it is critical to carefully design the rounding and lower bound functions to balance this trade-off.

Value Approximation. Given a rounding down function $\lfloor \cdot \rfloor_{\mathcal{G}}$, we would ideally use the rounded set $\left\{ \lfloor v \rfloor_{\mathcal{G}} \mid v \in \mathcal{V} \right\}$ to form our approximate state space. To avoid having to compute $\mathcal{V}$ explicitly, we instead use the rounded superset $\left\{ \lfloor v \rfloor_{\mathcal{G}} \mid v \in [v_{min}, v_{max}] \right\}$, where v_{min} and v_{max} are bounds on the extremal values that we specify later. To ensure we can use Algorithm 12 to find solutions efficiently, we must also relax the augmented action space to only include vectors that lead to feasible subproblems for (ADP). From Lemma 11, we know this is exactly the set of $(a, \hat{\mathbf{v}})$ for which $\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s,a)) \geq \kappa(v)$. Combining these ideas yields the new approximate MDP, defined in Definition 16.

Definition 16 (Approximate MDP)**.** Given a rounding function $\lfloor \cdot \rfloor_{\mathcal{G}}$ and lower bound function κ, the *approximate MDP* $\hat{M} \stackrel{\text{def}}{=} (\hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{P}, \hat{c}, H)$ where,

1. $\hat{\mathcal{S}} \stackrel{\text{def}}{=} \mathcal{S} \times \hat{\mathcal{V}}$ where $\hat{\mathcal{V}} \stackrel{\text{def}}{=} \left\{ \lfloor v \rfloor_{\mathcal{G}} \mid v \in [v_{min}, v_{max}] \right\}$.

2. $\hat{\mathcal{A}}_h(s,\hat{v}) \stackrel{\text{def}}{=} \left\{(a,\hat{\mathbf{v}}) \in \mathcal{A} \times \hat{\mathcal{V}}^S \mid \hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s,a)) \geq \kappa(\hat{v})\right\}$.

3. $\hat{P}_h((s',\hat{v}') \mid (s,\hat{v}),(a,\hat{\mathbf{v}})) \stackrel{\text{def}}{=} P_h(s' \mid s,a)[\hat{v}' = \hat{v}_{s'}]$.

4. $\hat{c}_h((s,\hat{v}),(a,\hat{\mathbf{v}})) \stackrel{\text{def}}{=} c_h(s,a)$.

The objective for $\hat{M}$ is to minimize the cost function $\hat{C} \stackrel{\text{def}}{=} C_{\hat{M}}$ with modified base case $\hat{C}^{\pi}_{H+1}(s,\hat{v}) \stackrel{\text{def}}{=} \chi_{\{\hat{v}\leq 0\}}$.

We can show that rounding down in Definition 16 achieves our goal of producing smaller cost policies. This ensures feasibility is even easier to achieve. We formalize this observation in Lemma 12.

Lemma 12 (Optimistic Costs)**.** *For our later choices of $\lfloor\cdot\rfloor_{\mathcal{G}}$ and κ, the following holds: for any $h \in [H+1]$ and $(s,v) \in \bar{\mathcal{S}}$, we have $\hat{C}^*_h(s, \lfloor v\rfloor_{\mathcal{G}}) \leq \bar{C}^*_h(s,v)$.*

Thus, Algorithm 13 always outputs a policy with better than optimal cost when the instance is feasible, $V^*_M > -\infty$. If the instance is infeasible, all policies have cost larger than B by definition and so Algorithm 13 correctly indicates the instance is infeasible. The remaining question is whether Algorithm 13 outputs policies having near-optimal value.

Time-Space Errors. To assess the optimality gap of Algorithm 13 policies, we must first explore the error accumulated by our rounding approach. Rounding each value naturally accumulates approximation error over time. Rounding the partial values while running Algorithm 11 accumulates additional error over (state) space. Thus, solving $\hat{M}$ using Algorithm 12 accumulates error over both time and space, unlike other approximate methods in RL. As a result, our rounding and threshold functions will generally depend on both H and S.

Algorithm 13 Approximation Scheme

Input: (M, C, B)

1: **Hyperparameters:** $\lfloor \cdot \rfloor_{\mathcal{G}}$ and κ
2: $\hat{M}, \hat{C} \leftarrow$ Definition 16$(M, C, \lfloor \cdot \rfloor_{\mathcal{G}}, \kappa)$
3: $\pi, \hat{C}^* \leftarrow$ Algorithm 12$(\hat{M}, \hat{C})$
4: **if** $\hat{C}_1^*(s_0, \hat{v}) > B$ for all $\hat{v} \in \hat{\mathcal{V}}$ **then**
5: **return** "Infeasible"
6: **else**
7: **return** π

Arithmetic Rounding. Our first approach is to round each value down to its closest element in a δ-cover. This guarantees that $v - \delta \leq \lfloor v \rfloor_{\mathcal{G}} \leq v$. Thus, $\lfloor v \rfloor_{\mathcal{G}}$ is an underestimate that is not too far from the true value. By setting δ to be inversely proportional to SH, we control the errors over time and space. The lower bound must also be a function of S since it controls the error over space.

Definition 17 (Additive Approx)**.** Fix $\epsilon > 0$. We define,

$$\lfloor v \rfloor_{\mathcal{G}} \stackrel{\text{def}}{=} \left\lfloor \frac{v}{\delta} \right\rfloor \delta \text{ and } \kappa(v) \stackrel{\text{def}}{=} v - \delta(S+1), \tag{5.6}$$

where $\delta \stackrel{\text{def}}{=} \frac{\epsilon}{H(S+1)+1}$, $v_{min} \stackrel{\text{def}}{=} -Hr_{max}$, and $v_{max} \stackrel{\text{def}}{=} Hr_{max}$.

Theorem 12 (Additive FPTAS)**.** *For any $\epsilon > 0$, Algorithm 13 using Definition 17 given any cMDP M and TSR criteria C either correctly outputs the instance is infeasible, or produces a policy π satisfying $\hat{V}^\pi \geq V_M^* - \epsilon$ in $O(H^7 S^5 A r_{max}^3 / \epsilon^3)$ time. Thus, it is an additive-FPTAS for the class of cMDPs with polynomial-bounded r_{max} and TSR criteria.*

Geometric Rounding. Since the arithmetic approach can be slow when r_{max} is large, we can instead round values down to their closest power of $1/(1-\delta)$. This guarantees the number of approximate values needed is upper bounded by a function of $\log(r_{max})$, which is polynomial in the input size. We choose a geometric scheme

satisfying $v(1-\delta) \leq \lfloor v \rfloor_{\mathcal{G}} \leq v$ so that the rounded value is an underestimate and a relative approximation to the true value. To ensure this property, we must now require that all rewards are non-negative.

Definition 18 (Relative Approx)**.** Fix $\epsilon > 0$. We define,

$$\lfloor v \rfloor_{\mathcal{G}} \stackrel{\text{def}}{=} v^{min} \left(\frac{1}{1-\delta} \right)^{\left\lfloor \log_{\frac{1}{1-\delta}} \frac{v}{v^{min}} \right\rfloor} \text{ and } \kappa(v) \stackrel{\text{def}}{=} v(1-\delta)^{S+1}, \tag{5.7}$$

where $\delta \stackrel{\text{def}}{=} \frac{\epsilon}{H(S+1)+1}$, $v_{min} = p_{min}^{H} r_{min}$, and $v_{max} = H r_{max}$.

Theorem 13 (Relative FPTAS)**.** *For $\epsilon > 0$, Algorithm 13 using Definition 18 given any cMDP M and TSR criteria C either correctly outputs the instance is infeasible, or produces a policy π satisfying $\hat{V}^{\pi} \geq V_M^*(1-\epsilon)$ in $O(H^7 S^5 A \log(r_{max}/r_{min} p_{min})^3 / \epsilon^3)$ time. Thus, it is a relative-FPTAS for the class of cMDPs with non-negative rewards and TSR criteria.*

Remark 15 (Assumption Necessity). We also note the mild reward assumptions we made to guarantee efficiency are unavoidable. Without reward bounds, (CON) captures the knapsack problem which does not admit additive approximations. Similarly, without non-negativity, relative approximations for maximization problems are generally not computable.

5.6 Conclusions

In this chapter, we studied the computational complexity of computing deterministic policies for CRL problems. Our main contribution was the design of an FPTAS, Algorithm 13, that solves (CON) for any cMPD and TSR criteria under mild reward assumptions. In particular, our method is an additive-FPTAS if the cMDP's rewards

are polynomially bounded, and is a relative-FPTAS if the cMDP's rewards are non-negative. We note these assumptions are necessary for efficient approximation, so our algorithm achieves the best approximation guarantees possible under worst-case analysis. Moreover, our algorithmic approach, which uses approximate dynamic programming over time and the state space, highlights the importance of the TSR condition in making (CON) tractable. Our work finally resolves the long-standing open questions of polynomial-time approximability for 1) anytime-constrained policies, 2) almost-sure-constrained policies, and 3) deterministic expectation-constrained policies.

Chapter 6

Bicriteria for Arbitrary Constraints

Abstract. In this chapter, we study the computational complexity of approximating general constrained Markov decision processes. Our primary contribution is the design of a polynomial time $(0, \epsilon)$-additive bicriteria approximation algorithm for finding optimal constrained policies across a broad class of recursively computable constraints, including almost-sure, chance, expectation, and their anytime variants. Matching lower bounds imply our approximation guarantees are optimal so long as $P \neq NP$. The generality of our approach results in answers to several long-standing open complexity questions in the constrained reinforcement learning literature. Specifically, we are the first to prove polynomial-time approximability for the following settings: policies under chance constraints, deterministic policies under multiple expectation constraints, policies under non-homogeneous constraints (i.e., constraints of different types), and policies under constraints for continuous-state processes.

6.1 Introduction

Constrained Reinforcement Learning (CRL) is growing increasingly crucial for managing complex, real-world applications such as medicine [26, 87, 64], disaster re-

lief [35, 117, 109], and resource management [75, 70, 90, 13]. Various constraints, including expectation [5], chance [123], almost-sure [18], and anytime constraints [78], were each proposed to address new challenges. Despite the richness of the literature, most works focus on stochastic, expectation-constrained policies, leaving many popular settings with longstanding open problems. Even chance constraints, arguably a close second in popularity, still lack any polynomial-time, even approximate, algorithms despite being introduced over a decade ago [123]. Other settings for which polynomial-time algorithms are open include deterministic policies under multiple expectation constraints, policies under non-homogeneous constraints (i.e., constraints of different types), and policies under constraints for continuous-state processes. Consequently, we study the computational complexity of general constrained problems to resolve many of these fundamental open questions.

Formally, we study the solution of *Constrained Markov Decision Processes* (CMDPs). Here, we define a CMDP through three fundamental parts: (1) an MDP M that accumulates both rewards and costs, (2) a general cost criterion C, and (3) a budget vector B. Additionally, we allow the agent to specify whether they require their policy to be deterministic or stochastic, formalized through a goal policy class $\overline{\Pi}$. The agent's goal is to solve $\max_{\pi \in \overline{\Pi}} V_M^\pi$ subject to $C_M^\pi \leq B$, where V_M^π denotes the agent's value and C_M^π denotes the agent's cost under π. This model can capture very general problems, including minimum time routes for self-driving vehicles that must satisfy 1) an anytime constraint on fuel consumption, 2) an expectation constraint on CO2 consumption, and 3) a chance constraint on vehicle wear and tear. Our main question is the following:

Can general CMDPs be approximated in polynomial time?

Hardness. Solving general CMDPs is notoriously challenging. When restricted to deterministic policies, solving a CMDP with just one constraint is NP-hard [36, 123, 78, 76]. This difficulty increases with the number of constraints: with at least two constraints, finding a feasible deterministic policy, let alone a near-optimal one, becomes NP-hard [78]. Even if we relax the deterministic requirement, this hardness remains for all constraint types other than expectation. Computational hardness aside, standard RL techniques fail to apply due to the combinatorial nature induced by many constraint types. Adding in additional constraints with fundamentally different structures further complicates the problem.

Past Work. A few works have managed to derive provable approximation algorithms for some cases of CRL. McMahan [76] presented a fully polynomial-time approximation scheme (FPTAS) for the computation of deterministic policies of a general class of constraints, which includes expectation, almost-sure, and anytime constraints. Although powerful, their framework only works for one constraint and fails to capture anytime-expectation constraints along with chance constraints. Similarly, Khonji et al. [63] achieves an FPTAS for expectation and chance constraints, but only in the constant horizon setting. In contrast, McMahan and Zhu [78] develops a polynomial-time $(0, \epsilon)$-additive bicritiera approximation algorithm for anytime and almost-sure constraints. However, their framework is specialized to those constraint types and thus fails for our purpose. In contrast, Xu and Mannor [123] developed a pseudo-polynomial time algorithm for finding feasible chance-constrained policies, but their methods do not lead to polynomial-time solutions.

Our Contributions. We design a polynomial-time $(0, \epsilon)$-additive bicriteria approximation algorithm for tabular, SR-criterion CMDPs. An SR criterion is required to satisfy a generalization of the policy evaluation equations and includes expectation,

chance, and almost-sure constraints along with their anytime equivalents. Our framework implies the first positive polynomial-time approximability results for (1) policies under chance constraints, (2) deterministic policies under multiple expectation constraints, and (3) policies under non-homogeneous constraints – each of which has been unresolved for over a decade. We then extend our algorithm into a polynomial-time (ϵ, ϵ)-additive bicriteria approximation algorithm for continuous-state CMPDs under a general class of constraints, which includes expectation, almost-sure, and anytime constraints.

Our Techniques. Our algorithm requires several key techniques. First, we transform a constraint concerning all realizable histories into a simpler per-time constraint. We accomplish this by augmenting the state space with an artificial budget and augmenting the action space to choose future budgets to satisfy the constraint. However, Bellman updates then become as hard as the knapsack problem due to the large augmented action space. For tabular cMDPs, we show that the Bellman updates can be approximately computed using dynamic programming and rounding. By strategically rounding the artificial budget space, we achieve a $(0, \epsilon)$-bicriteria approximation for tabular CMDPs. By appropriately discretizing the continuous state space, our method becomes a (ϵ, ϵ)-bicriteria approximation algorithm for continuous state CMDPs.

6.1.1 Related Work

Constrained RL. It is known that stochastic expectation-constrained policies are polynomial-time computable via linear programming [5], and many planning and learning algorithms exist for them [89, 110, 14, 53]. Some learning algorithms can even avoid violation during the learning process under certain assumptions [115, 7].

Furthermore, Brantley et al. [17] developed no-regret algorithms for cMDPs and extended their algorithms to the setting with a constraint on the cost accumulated over all episodes, which is called a knapsack constraint [17, 22].

Safe RL. The safe RL community [39, 45] has mainly focused on no-violation learning for stochastic expectation-constrained policies [24, 16, 4, 21, 10] and solving chance constraints [114, 132], which capture the probability of entering unsafe states. Performing learning while avoiding dangerous states [132] is a special case of expectation constraints that has also been studied [96, 107] under non-trivial assumptions. In addition, instantaneous constraints, which require the immediate cost to be within budget at all times, have also been studied [69, 37, 43].

6.2 Constraints

Cost-Accumulating MDPs. In this work, we consider environments that accumulate both rewards and costs. Formally, we consider a (finite-horizon, tabular) *cost-accumulating Markov Decision Process* (caMDP) tuple $M = (H, \mathcal{S}, \mathcal{A}, P, R, C, s_0)$, where (i) $H \in \mathbb{Z}_{\geq 0}$ is the finite time *horizon*, (ii) $\mathcal{S}_h$ is the finite set of *states*, (iii) $\mathcal{A}_h(s)$ is the finite set of available *actions*, (iv) $P_h(s, a) \in \Delta(S)$ is the *transition* distribution for a given state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ (note, $\Delta(S)$ represents the probability simplex over $\mathcal{S}$), (v) $R_h(s, a) \in \Delta(\mathbb{R})$ is the *reward* distribution, (vi) $C_h(s, a) \in \Delta(\mathbb{R}^m)$ is the *cost* distribution, and (vii) $s_0 \in \mathcal{S}$ is the initial state.

To simplify notation, we let $r_h(s, a) \stackrel{\text{def}}{=} \mathbb{E}[R_h(s, a)]$ denote the expected reward, $c_h(s, a)$ represent the cost function when costs are deterministic (which will be the case throughout the main text), $S \stackrel{\text{def}}{=} |\mathcal{S}|$ denote the number of states, $A \stackrel{\text{def}}{=} |\mathcal{A}|$ denote the number of joint actions, $[H] \stackrel{\text{def}}{=} \{1, \ldots, H\}$, $\mathcal{M}$ be the set of all caMDPs, and $|M|$ be the total description size of the caMDP. We also use the Iverson Bracket

notation $[P] \stackrel{\text{def}}{=} 1_{\{P=True\}}$ and the characteristic function χ_P which is ∞ when P is False and 0 otherwise.

Agent Interactions. The agent interacts with M using a *policy* $\pi = \{\pi_h\}_{h=1}^{H}$. In the fullest generality, $\pi_h : \mathcal{H}_h \to \Delta(\mathcal{A})$ is a mapping from the observed history at time h (including costs) to a distribution of actions. Often, researchers study *Markovian policies*, which take the form $\pi_h : \mathcal{S} \to \Delta(\mathcal{A})$, and *deterministic policies*, which take the form $\pi_h : \mathcal{H}_h \to \mathcal{A}$. We let Π denote the set of all policies and Π^D denote the set of all deterministic policies.

The agent starts in the initial state s_0 with observed history $\tau_1 = (s_0)$. For any $h \in [H]$, the agent chooses a joint action $a_h \sim \pi_h(\tau_h)$. Then, the agent receives immediate reward $r_h \sim R_h(s, a)$ and cost vector $c_h \sim C_h(s, a)$. Lastly, M transitions to state $s_{h+1} \sim P_h(s_h, a_h)$, prompting the agent to update its observed history to $\tau_{h+1} = (\tau_h, a_h, c_h, s_{h+1})$. This process is repeated for H steps; the interaction ends once s_{H+1} is reached.

Constrained Processes. Suppose the agent has a *cost criterion* $C : \mathcal{M} \times \Pi \to \mathbb{R}^m$ and a corresponding *budget* vector $B \in \mathbb{R}^m$. We refer to the tuple (M, C, B) as a *Constrained Markov Decision Process* (CMDP). Given a CMDP and desired policy class $\overline{\Pi} \in \{\Pi^D, \Pi\}$, the agent's goal is to solve the constrained optimization problem:

$$\begin{aligned} \max_{\pi \in \overline{\Pi}} \; & V_M^\pi \\ \text{s.t. } & C_M^\pi \leq B \end{aligned} \tag{CON}$$

In the above, $V_M^\pi \stackrel{\text{def}}{=} \mathbb{E}_M^\pi \left[\sum_{h=1}^{H} r_h(s_h, a_h) \right]$ denotes the value of a policy π, $\mathbb{E}_M^\pi$ denotes the expectation defined by $\mathbb{P}_M^\pi$, and $\mathbb{P}_M^\pi$ denotes the probability law over histories induced from the interaction of π with M. Lastly, we let V^* denote the

optimal solution value to (CON). In the main paper, we focus on the case where $\overline{\Pi} = \Pi^D$.

SR Criteria. We study cost criteria that generalize the standard policy evaluation equations to enable dynamic programming techniques. In particular, we require the cost of a policy to be recursively computable with respect to the time horizon. For our later approximations in Section 6.5, we will also need key functions defining the recursion to be short maps, i.e., 1-Lipschitz, with respect to the infinity norm.

Definition 19 (SR)**.** We call a cost criterion *shortly recursive* (SR) if for any caMDP M and any policy $\pi \in \Pi^D$, π's cost decomposes recursively into $C_M^\pi = C_1^\pi(s_0)$, where $C_{H+1}^\pi = 0$ and for all $h \in [H]$ and $\tau_h \in \mathcal{H}_h$ letting $s = s_h(\tau_h)$ and $a = \pi_h(\tau_h)$,

$$C_h^\pi(\tau_h) = c_h(s,a) + \underset{s'}{f}\, g\left(P_h\left(s' \mid s,a\right)\right) C_{h+1}^\pi\left(\tau_h, a, s'\right). \tag{SR}$$

Here, $f_{s'}$ is the finite extension of an associative, non-decreasing, binary function f, and g is a $[0,1]$-valued function rooted at 0. Moreover, we require that f is a short map when either of its inputs are fixed, satisfies $f(0,x) = f(x,0) = x$ for all x, and when combined with g, i.e., $f_{s'}\, g\left(P_h\left(s' \mid s,a\right)\right) x_{s'}$, is a short map in x.

Remark 16 (Stochastic Variants). Our results generalize to both stochastic policies and stochastic costs as well. The algorithmic approach is identical, but the definitions and analysis are more complex. Consequently, we focus on the deterministic cases in the main text.

Constraint Formulations. The fundamental constraints considered in the CRL literature are Expectation, Chance, and Almost-sure constraints. Each of these induces a natural *anytime* variant that stipulates the required constraint must hold for the truncated cost $\sum_{h=1}^{t} c_h$ at all times $h \in [H]$. We give the formal definitions

Con/Time	Expectation	Chance	Almost-Sure
Classical	$\mathbb{E}^{\pi}_{M}\left[\sum_{h=1}^{H} c_h\right] \leq B$	$\mathbb{P}^{\pi}_{M}\left[\sum_{h=1}^{H} c_h > B\right] \leq \delta$	$\mathbb{P}^{\pi}_{M}\left[\sum_{h=1}^{H} c_h \leq B\right] = 1$
$\forall t \in [H]$	$\mathbb{E}^{\pi}_{M}\left[\sum_{h=1}^{t} c_h\right] \leq B$	$\mathbb{P}^{\pi}_{M}\left[\sum_{h=1}^{t} c_h > B\right] \leq \delta$	$\mathbb{P}^{\pi}_{M}\left[\sum_{h=1}^{t} c_h \leq B\right] = 1$

Figure 6.1: The Constraint Landscape

in Figure 6.1. Importantly, each constraint is equivalent to $C^{\pi}_{M} \leq B'$ for some appropriately chosen SR criteria.

Proposition 17 (SR Modeling)**.** *The classical constraints can be modeled by SR constraints of the form $C^{\pi}_{M} \leq B'$ as follows:*

1. *Expectation Constraints – $f(x,y) \stackrel{def}{=} x+y$, $g(x) \stackrel{def}{=} x$, and $B' \stackrel{def}{=} B$.*
2. *Chance Constraints – (f,g) defined as above and $B' \stackrel{def}{=} \delta$. Here, we assume M's states are augmented with cumulative costs and that $c_h((s,\bar{c}),a) \stackrel{def}{=} [c_h(s,a)+\bar{c} > B]$ for the anytime variant and $c_h((s,\bar{c}),a) \stackrel{def}{=} [c_h(s,a)+\bar{c} > B][h = H]$ otherwise.*
3. *Almost-sure Constraints – $f(x,y) \stackrel{def}{=} \max(x,y)$, $g(x) \stackrel{def}{=} [x > 0]$, and $B' \stackrel{def}{=} B$. Anytime variant – $f(x,y) \stackrel{def}{=} \max(0, \max(x,y))$ while g and B' remain the same.*

General anytime variants, including anytime expectation constraints, can be modeled by $\left\{C^{\pi}_{M,t} \leq B\right\}_{t\in[H]}$ where $C^{\pi}_{M,t}$ is the original SR criterion but defined for the truncated-horizon process with horizon t.

Computational Limitations. It is known that computing feasible policies for CMDPs is NP-hard [76, 78]. As such, we must relax feasibility for any hope of polynomial-time algorithms. Consequently, we focus on *bicriteria* approximation algorithms.

Definition 20 (Bicriteria)**.** A policy π is an (α, β)-*additive bicriteria approximation* to a CMDP (M, C, B) if $V_M^\pi \geq V^* - \alpha$ and $C_M^\pi \leq B + \beta$. We refer to an algorithm as an (α, β)-*bicriteria* if for any CMDP it outputs an (α, β)-*additive bicriteria approximation* or correctly reports the instance is infeasible.

The existence of a polynomial-time bicriterion for our general constrained problem implies brand-new approximability results for many classic problems in the CRL literature. For clarity, we will explicitly state the complexity-theoretic implications for each classical setting.

Theorem 14 (Implications)**.** *A polynomial-time* (ϵ, ϵ)*-bicriteria implies that in polynomial time it is possible to compute a policy* $\pi \in \overline{\Pi}$ *satisfying* $V_M^\pi \geq V^* - \epsilon$ *and any constant combination of the following constraints:*

1. $\mathbb{E}_M^\pi \left[\sum_{h=1}^H c_h\right] \leq B + \epsilon$
2. $\mathbb{P}_M^\pi \left[\sum_{h=1}^H c_h \leq B + \epsilon\right] = 1$
3. $\mathbb{P}_M^\pi \left[\sum_{h=1}^H c_h > B + \epsilon\right] \leq \delta + \epsilon.$

In other words, polynomial-time approximability is possible for each of the settings described in Section 6.1 when the number of constraints is constant.

Remark 17 (Extensions)*.* All of our results hold for Markov games and infinite discounted settings.

6.3 Reduction

In this section, we present a general solution approach to SR-criterion CMDPs. Our approach revolves around converting the general cost constraint into a per-step action constraint. This leads to the design of an augmented MDP that can be solved with standard RL methods.

Augmentation. State augmentation is the known approach for solving anytime-constrained MDPs [78]. The augmentation permits the problem to be solved by the following dynamic program:

$$V_h^*(s,c) = \max_{\substack{a\in\mathcal{A},\\ c+c_h(s,a)\leq B}} r_h(s,a) + \sum_{s'} P_h(s' \mid s,a) V_{h+1}^*(s, c + c_h(s,a)). \tag{6.1}$$

When moving to other constraints, the cumulative cost may no longer suffice to determine constraint satisfaction. For example, the expected cost depends on the cumulative cost of all realizable branches, not just the current branch.

Expectation Constraints. Instead, we can exploit the recursive nature of the expected cost to find a solution. Suppose at stage (s,h) we impose an artificial budget b on the expected cost of a policy π from time h onward: $\mathbb{E}^\pi\left[\sum_{t=h}^H c_t\right] \leq b$. By the policy evaluation equations, we know this equates to satisfying:

$$C_h^\pi(s) = c_h(s,a) + \sum_{s'} P_h(s' \mid s,a) C_{h+1}^\pi(s') \leq b. \tag{6.2}$$

For this invariant to be satisfied, it suffices for the agent to choose future artificial budgets $b_{s'}$ for $s' \in \mathcal{S}$ satisfying,

$$c_h(s,a) + \sum_{s'} P_h(s' \mid s,a) b_{s'} \leq b. \tag{6.3}$$

and ensure the future artificial budgets are obeyed inductively: $C_{h+1}^\pi(s', b_{s'}) \leq b_{s'}$.

General Approach. We can apply the same reasoning for general recursively computable cost criteria. If C is SR, then we know that $C_h^\pi(s)$ obeys (SR). Thus, to

guarantee that $C_h^\pi(s) \leq b$ it suffices to choose $b_{s'}$'s satisfying,

$$c_h(s,a) + \underset{s'}{f}\, g\left(P_h(s' \mid s,a)\right) b_{s'} \leq b, \tag{6.4}$$

and inductively guarantee that $C_{h+1}^\pi(s') \leq b_{s'}$.

We can view choosing future artificial budgets as part of the agent's augmented actions. Then, at any augmented state (s,b), the agent's augmented action space includes all $(a,\mathbf{b}) \in \mathcal{A} \times \mathbb{R}^S$ satisfying (6.3). When M transitions to $s' \sim P_h(s,a)$, the agent's new augmented state should consist of the environment's new state in addition to its chosen demand for that state, $(s', b_{s'})$. Putting these pieces together yields the definition of the reduced, action-constrained MDP, Definition 21.

Definition 21 (Reduced MDP)**.** Given any SR-criterion CMDP (M, C, B), we define the *reduced MDP* $\bar{M} \overset{\text{def}}{=} (H, \bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{P}, \overline{R}, \bar{s}_0)$ where,

1. $\bar{\mathcal{S}}_h \overset{\text{def}}{=} \mathcal{S}_h \times \mathcal{B}$ where $\mathcal{B} \overset{\text{def}}{=} \bigcup_{\pi \in \Pi^D} \bigcup_{h \in [H+1]} \bigcup_{\tau_h \in \mathcal{H}_h} \{C_h^\pi(\tau_h)\}$
2. $\bar{\mathcal{A}}_h(s,b) \overset{\text{def}}{=} \{(a,\mathbf{b}) \in \mathcal{A}_h(s) \times \mathbb{R}^S \mid c_h(s,a) + f_{s'}\, g\left(P_h(s' \mid s,a), b_{s'}\right) \leq b\}$
3. $\bar{P}_h((s',b') \mid (s,b),(a,\mathbf{b})) \overset{\text{def}}{=} P_h(s' \mid s,a)[b' = b_{s'}]$
4. $\overline{R}_h((s,b),(a,\mathbf{b})) \overset{\text{def}}{=} R_h(s,a)$
5. $\bar{s}_0 \overset{\text{def}}{=} (s_0, B)$

We also re-define the base case value to $\bar{V}_{H+1}^*(s,b) \overset{\text{def}}{=} -\chi_{\{b \geq 0\}}$. Note, the reduced MDP has a non-stationary state and action set, unlike the base MDP.

Reduction. Importantly, $\bar{M}$'s augmented action space ensures constraint satisfaction. Thus, we have effectively reduced a problem involving total history constraints to one with only standard per-time-step constraints. So long as our cost is SR, $\bar{M}$

Algorithm 14 Reduction

Input: (M, C, B)

$\bar{M} \leftarrow$ Definition 21(M, C, B)
$\pi, \bar{V}^* \leftarrow \textsc{Solve}(\bar{M})$
if $\bar{V}^* = -\infty$ **then**
    return "Infeasible"
else
    return π

Algorithm 15 Augmented Interaction

Input: π

$\bar{s}_1 = (s_0, B)$
for $h \leftarrow 1$ to H **do**
    $(a, \mathbf{b}) \leftarrow \pi_h(\bar{s}_h)$
    $s_{h+1} \sim P_h(s_h, a)$
    $\bar{s}_{h+1} = (s_{h+1}, b_{s_{h+1}})$

can be solved using fast RL methods instead of the brute force computation required for general CMDPs. These properties ensure our method, Algorithm 14, is correct.

Lemma 13 (Value)**.** *For any $h \in [H+1]$, $\tau_h \in \mathcal{H}_h$, and $b \in \mathcal{B}$, if $s = s_h(\tau_h)$, then,*

$$\bar{V}_h^*(s, b) \geq \sup_{\pi \in \Pi^D} V_h^\pi(\tau_h) \quad s.t.\ C_h^\pi(\tau_h) \leq b. \tag{6.5}$$

Lemma 14 (Cost)**.** *Suppose that $\pi \in \Pi^D$. For all $h \in [H+1]$ and $(s, b) \in \bar{\mathcal{S}}$, if $\bar{V}_h^\pi(s, b) > -\infty$, then $\bar{C}_h^\pi(s, b) \leq b$.*

Theorem 15 (Reduction)**.** *If* Solve *is any finite-time MDP solver, then Algorithm 14 correctly solves* (CON) *in finite time for any SR-criterion CMDP.*

Remark 18 (Deployment). Given a budget-augmented policy π output from Algorithm 14, the agent can execute π using Algorithm 15.

6.4 Bellman Updates

In this section, we discuss efficient methods for solving $\bar{M}$. Our approach relies on using (SR) to break down the Bellman update so that it is solvable using dynamic programming. We then use dynamic rounding to achieve an efficient approximation algorithm.

Bellman Hardness. Even a small set of artificial budgets, $\mathcal{B}$, needed to be considered, solving $\bar{M}$ would still be challenging due to its exponentially large, constrained action space. Just one Bellman update equates to solving the constrained optimization problem:

$$\begin{aligned} \bar{V}_h^*(s,b) = \max_{a,\mathbf{b}} \; & r_h(s,a) + \sum_{s'} P_h(s' \mid s,a) V_{h+1}^*(s', b_{s'}) \\ \text{s.t. } & c_h(s,a) + \underset{s'}{f}\, g\left(P_h\left(s' \mid s,a\right)\right) b_{s'} \leq b. \end{aligned} \tag{BU}$$

Above, we used the fact that $(s', b') \in \text{Supp}(\bar{P}_h((s,b),(a,\mathbf{b})))$ iff $s' \in \text{Supp}(P_h(s,a))$ and $b' = b_{s'}$. In fact, even when each $b_{s'}$ only takes on two possible values, $\{0, w_{s'}\}$, this optimization problem generalizes the knapsack problem, implying that it is NP-hard to solve.

Dynamic Programming. To get around this computational bottleneck, we must fully exploit Definition 19. For any fixed $(h,(s,b),a)$, the key idea is to treat choosing b's as its own sequential decision-making problem. Suppose we have already chosen $b_1, \ldots, b_{t-1}$ leading to partial cost $F \overset{\text{def}}{=} f_{s'=1}^{t-1} g(P_h(s' \mid s,a)) b_{s'}$. Since f is associative, we can update our partial cost after choosing b_t to $f(F, g(P_h(t \mid s,a))b_t)$. Once we have made a choice for each future state, we can verify if $(a,\mathbf{b}) \in \bar{\mathcal{A}}_h(s,b)$ by checking the condition: $c_h(s,a) + F \leq b$. By incorporating the value objective, we design a

dynamic program for computing (BU).

Definition 22 (DP)**.** For any $h \in [H]$, $(s,b) \in \bar{\mathcal{S}}$, $a \in \mathcal{A}$ and $F \in \mathbb{R}$, we define $\bar{V}^{s,a}_{h,b}(S+1,F) = -\chi_{\{c_h(s,a)+F \le b\}}$, and for any $t \in [S]$,

$$\bar{V}^{s,a}_{h,b}(t,F) \stackrel{\text{def}}{=} \max_{b_t \in \mathcal{B}} P_h(t \mid s,a)\bar{V}^*_{h+1}(t,b_t) + \bar{V}^{s,a}_{h,b}\left(t+1, f\left(F, g(P_h(t \mid s,a))b_t\right)\right). \quad (6.6)$$

Lemma 15 (DP Correctness)**.** *For any $h \in [H]$ and $(s,b) \in \bar{\mathcal{S}}$, we have that* $\bar{V}^*_h(s,b) = \max_{a\in\mathcal{A}} r_h(s,a) + \bar{V}^{s,a}_{h,b}(1,0)$.

Dynamic Rounding. Although a step in the right direction, solving Definition 22 can still be slow due to the exponential number of considered partial costs. We resolve this issue by rounding each partial cost to an element of some small set $\hat{\mathcal{F}}$. Since f need not be linear, using rounding in a preprocessing step does not suffice: we must re-round at each step to ensure inputs are a valid element of our input set.

For any $\ell > 0$, we view ℓ as a new unit length. Our rounding function maps any real number to its closest upper bound in the set of integer multiples of ℓ. We use upper bounds to guarantee that the rounded partial costs are always larger than the true partial costs. Smaller ℓ ensures less approximation error, while larger ℓ ensures fewer considered partial costs. Thus, ℓ directly controls the accuracy-efficiency trade-off.

Definition 23 (Rounding Functions)**.** For any $\ell > 0$ and $x \in \mathbb{R}$, we define $\lceil x \rceil_\ell \stackrel{\text{def}}{=} \left\lceil \frac{x}{\ell} \right\rceil \ell$ to be the smallest integer multiple of ℓ that is larger than x. We also define $\kappa_\ell(x) \stackrel{\text{def}}{=} x + \ell(S+1)$. Note, when considering vectors, all operations are performed component-wise.

Since we round up the partial costs, the approximate partial cost of a feasible $\mathbf{b}$ could exceed b. To ensure all feasible choices of $\mathbf{b}$ are considered, we must also

relax the budget comparison. Instead, we compare partial costs to a carefully chosen upper threshold $\kappa(b)$. Putting these pieces together yields our approximate Bellman update method.

Definition 24 (Approximate Update)**.** Fix any $\ell > 0$ and function $\kappa : \mathbb{R}^m \to \mathbb{R}^m$. For any $h \in [H]$, $(s,b) \in \bar{\mathcal{S}}$, $a \in \mathcal{A}$ and $\hat{F} \in \mathbb{R}^m$, we define $\hat{V}^{s,a}_{h,b}(S+1, \hat{F}) \stackrel{\text{def}}{=} -\chi_{\left\{c_h(s,a)+\hat{F} \leq \kappa(b)\right\}}$, and for any $t \in [S]$,

$$\hat{V}^{s,a}_{h,b}(t, \hat{F}) \stackrel{\text{def}}{=} \max_{b_t \in \mathcal{B}} P_h(t \mid s,a)\bar{V}^*_{h+1}(t, b_t) + \hat{V}^{s,a}_{h,b}\left(t+1, \left\lceil f\left(\hat{F}, g(P_h(t \mid s,a))b_t\right)\right\rceil_\ell\right). \tag{ADP}$$

We then define the *approximate update* by,

$$\hat{V}^*_h(s,b) \stackrel{\text{def}}{=} \max_{a \in \mathcal{A}} r_h(s,a) + \hat{V}^{s,a}_{h,b}(1,0). \tag{AU}$$

Overall, solving the ADP yields an approximate solution.

Lemma 16 (Approximation)**.** *For any $h \in [H]$, $(s,b) \in \bar{\mathcal{S}}$, $a \in \mathcal{A}$, $\hat{F} \in \mathbb{R}^m$, and $t \in [S+1]$, we have that,*

$$\begin{aligned} \hat{V}^{s,a}_{h,b}(t, \hat{F}) = \max_{\mathbf{b} \in \mathcal{B}^{S-t+1}} & \sum_{s'=t}^{S} P_h(s' \mid s,a)\bar{V}^*_{h+1}(s', b_{s'}) \\ s.t. \quad & c_h(s,a) + \hat{f}^{s,a}_{h,\mathbf{b}}(t, \hat{F}) \leq \kappa(b), \end{aligned} \tag{6.7}$$

where $\hat{f}^{s,a}_{h,\mathbf{b}}(t, \hat{F})$ is the dynamic rounding of $f\left(\hat{F}, f^S_{s'=t}\, g(P_h(t \mid s,a), b_t)\right)$. Moreover, if $\lceil\rceil_\ell$ and κ are replaced with the identity function, (AU) is equivalent to (BU).

Remark 19 (DP details). Technically, to turn this recursion into a true dynamic program, we must also precompute the inputs to any subproblem. Unlike in standard RL, this computation must be done with a forward recursion. If we let $\hat{\mathcal{F}}^{s,a}_h(t)$

Algorithm 16 Approximate Backward Induction

Input: $\bar{M}$

$\hat{V}^*_{H+1}(s,b) \leftarrow \chi_{\{b \geq 0\}}$ for all $(s,b) \in \bar{\mathcal{S}}$
for $h \leftarrow H$ down to 1 **do**
  for $(s,b) \in \bar{\mathcal{S}}$ **do**
    $\hat{a}, \hat{V}^*_h(s,b) \leftarrow$ (AU)
    $\pi_h(s,b) \leftarrow \hat{a}$
return $\pi, \hat{V}^*$

denote the set of possible input rounded partial costs for state t, then the set satisfies the inductive relationship $\hat{\mathcal{F}}^{s,a}_h(1) \stackrel{\text{def}}{=} \{0\}$ and for any $t \in [S]$, $\hat{\mathcal{F}}^{s,a}_h(t+1) \stackrel{\text{def}}{=} \bigcup_{b_t \in \mathcal{B}} \bigcup_{F \in \hat{\mathcal{F}}^{s,a}_h(t)} \{\lceil f(F, + g(P_h(t \mid s,a))b_t \rceil_\ell\}$. This relationship translates directly into an iterative algorithm for computing all needed inputs. Using this gives a complete DP algorithm for solving (ADP)[1].

Theorem 16 (Approx Solve)**.** *When $\lceil \rceil_\ell$ and κ are replaced with the identity function, Algorithm 16 correctly solves any $\bar{M}$ produced from Definition 21. Moreover, Algorithm 16 runs in time $O\left(H^{m+1} S^{m+2} A |\mathcal{B}|^2 \left\| c_{max} - c_{min} \right\|^m_\infty / \ell^m\right)$.*

6.5 Bicriteria

Algorithm 16 allows us to approximately solve $\bar{M}$ in finite cases much faster than traditional methods. However, when $|\mathcal{B}|$ is large, the algorithm still runs in exponential time. Similarly to the partial cost rounding in Definition 24, we can reduce the size of $|\mathcal{B}|$ by considering a smaller approximate set based on rounding. Since we still desire optimistic budgets, we use the same rounding function from Definition 23 but with a different choice of ℓ.

[1] We use the notation $x, o \leftarrow \min_x z(x)$ to say that x is the minimizer and o the value of the optimization.

Budget Rounding. Rounding naturally impacts the state space, but has other consequences as well. To avoid complex computation, we consider the approximate set $\hat{\mathcal{B}} \stackrel{\text{def}}{=} \{\lceil b \rceil_\ell \mid b \in [b_{min}, b_{max}]\}$ where $[b_{min}, b_{max}] \supseteq \mathcal{B}$ is a superset of all required artificial budgets that we formalize later. As before, rounding the budgets may cause originally feasible choices to now violate the constraint. To ensure all feasible choices are considered and that we can use Algorithm 16 to get speed-ups, we define the approximate action space to include all vectors that lead to feasible subproblems of (ADP). From Lemma 16, we know this set is exactly the set of $(a, \hat{\mathbf{b}})$ satisfying $c_h(s,a) + \hat{f}^{s,a}_{h,\hat{\mathbf{b}}}(1,0) \leq \kappa(\hat{b})$. Putting these ideas together yields a new, approximate MDP.

Definition 25 (Approximate MDP)**.** Given any SR-criterion CMDP (M, C, B), we define the *approximate MDP* $\hat{M} \stackrel{\text{def}}{=} (H, \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{P}, \hat{R}, \hat{s}_0)$ where,

1. $\hat{\mathcal{S}}_h \stackrel{\text{def}}{=} \mathcal{S}_h \times \hat{\mathcal{B}}$ where $\hat{\mathcal{B}} \stackrel{\text{def}}{=} \{\lceil b \rceil_\ell \mid b \in [b_{min}, b_{max}]\}$.
2. $\hat{\mathcal{A}}_h(s, \hat{b}) \stackrel{\text{def}}{=} \{(a, \hat{\mathbf{b}}) \in \mathcal{A}_h(s) \times \hat{\mathcal{B}}^S \mid c_h(s,a) + \hat{f}^{s,a}_{h,\hat{\mathbf{b}}}(1,0) \leq \kappa(\hat{b})\}$
3. $\hat{P}_h((s', \hat{b}') \mid (s,b), (a, \hat{\mathbf{b}})) \stackrel{\text{def}}{=} P_h(s' \mid s, a)[\hat{b}' = \hat{b}_{s'}]$
4. $\hat{R}_h((s, \hat{b}), a) \stackrel{\text{def}}{=} R_h(s,a)$
5. $\hat{s}_0 \stackrel{\text{def}}{=} (s_0, \lceil B \rceil_\ell)$

We again re-define the base case value to $\hat{V}^*_{H+1}(s, \hat{b}) \stackrel{\text{def}}{=} -\chi_{\{\hat{b} \geq 0\}}$.

Since we always round budgets up, the agent can make even better choices than originally. It is then easy to see that policies for $\hat{M}$ always achieve optimal constrained value. We formalize this observation in Lemma 17.

Lemma 17 (Optimal Value)**.** *For any $h \in [H+1]$ and $(s,b) \in \bar{\mathcal{S}}$, $\hat{V}^*_h(s, \lceil b \rceil_\ell) \geq \bar{V}^*_h(s,b)$.*

Algorithm 17 Bicriteria

Input: (M, C, B)

Hyperparameter: ℓ
$\hat{M} \leftarrow$ Definition 25$(M, (f, g), B, \ell)$
$\pi, \hat{V}^* \leftarrow$ Algorithm 16$(\hat{M}, (f, g), \ell)$
if $\hat{V}_1^*(s_0, \lceil B \rceil_\ell) = -\infty$ **then**
    return "Infeasible"
else
    return π

Time-Space Errors. To assess the violation gap of Algorithm 17 policies, we must first explore the error accumulated by our rounding approach. Rounding each artificial budget naturally accumulates approximation error over time. Rounding the partial costs while running Algorithm 16 accumulates additional error over (state) space. Thus, solving $\hat{M}$ using Algorithm 17 accumulates error over both time and space, unlike standard approximate methods in RL. As a result, our rounding and threshold functions will generally depend on both H and S.

Arithmetic Rounding. Our approach is to round each value down to its closest element in an ℓ-cover. Using the same rounding as in Definition 23, we guarantee that $b \leq \lceil b \rceil_\ell \leq b + \ell$. Thus, $\lceil b \rceil_\ell$ is an overestimate that is not too far from the true value. By setting ℓ to be inversely proportional to SH, we control the errors over time and space. The lower bound must also be a function of S since it controls the error over space.

Lemma 18 (Approximate Cost)**.** *Suppose that $\pi \in \Pi^D$. For all $h \in [H+1]$ and $(s, \hat{b}) \in \hat{\mathcal{S}}$, if $\hat{V}_h^\pi(s, \hat{b}) > -\infty$, then $\hat{C}_h^\pi(s, \hat{b}) \leq \hat{b} + \ell(S+1)(H-h+1)$.*

Theorem 17 (Bicriteria)**.** *For any SR-criterion CMDP with polynomially-bounded costs and $\epsilon > 0$, the choice of $\ell \stackrel{def}{=} \frac{\epsilon}{1+(S+1)H}$ ensures Algorithm 17 is a $(0, \epsilon)$-bicriteria running in polynomial time $O\left(H^{6m+1} S^{4m+2} A \left\|c_{max} - c_{min}\right\|_\infty^{3m} / \epsilon^{3m}\right)$.*

Corollary 6 (Relative). *For any $\epsilon > 0$, the choice of $\ell \stackrel{def}{=} \frac{\epsilon}{B(H(S+1)+1)}$ ensures Algorithm 17 is a polynomial time $(0, 1+\epsilon)$-relative bicriteria for the class of polynomial-budget-bounded-cost CMDPs with SR-cost criteria. This includes all SR-criterion CMDPs with non-negative costs.*

Remark 20 (Chance Constraints). Technically, for chance constraints, we first create a cost-augmented MDP that is initially passed into the input. This allows us to write chance constraints in the SR form. Consequently, the S term in Theorem 17 is really a larger augmented S. To achieve ϵ cost violation, [78] showed that an augmented space of size $O(SH^2 \|c_{max} - c_{min}\|_\infty / \epsilon)$ is needed, which still results in a polynomial-time complexity.

Remark 21 (Approximation Optimality). [78] showed that our assumptions on cost bounds are necessary to achieve polynomial-time approximations. Thus, our approximation guarantees are the best possible. Moreover, we can show that our dependency on the number of constraints is also unavoidable. This is formalized in Proposition 18.

Proposition 18 (Multi-Constraint Hardness). *If $m = \Omega(n^{1/d})$ for some constant d, then computing an ϵ-feasible policy for a CMDP is NP-hard for any $\epsilon > 0$.*

6.5.1 Continuous MDPs

We also show that approximations are possible in infinite state settings under certain continuity assumptions.

Assumption 7 (Continuity). We assume the caMDP M is Lipschitz continuous. Formally, we require that (1) $S = [s_{min}, s_{max}]$, (2) the reward function is λ_r Lipschitz, (3) the cost function is λ_c Lipschitz, (4) the transitions are λ_p Lipschitz – each with respect to the state input, and (5) each of these quantities is polynomial-sized in the

input representation. For SR-criterion CMDPs, we also assume that f has a natural finite equivalent denoted $\tilde{f}$, g is a sublinear short map, and $f_{s'} z \leq (s_{max} - s_{min})$ for any constant z.

All we need to do is discretize the state space, and run our previous algorithm on the following discretized CMDP.

Definition 26 (Discretized CMDP)**.** Given any SR-criterion CMDP (M, C, B), we define the *discretized CMDP* $(\tilde{M}, \tilde{C}, B)$ where $\tilde{M} = (H, \tilde{\mathcal{S}}, \mathcal{A}, \tilde{P}, R, C, \tilde{s}_0)$ is the discretized caMDP defined by,

1. $\tilde{\mathcal{S}}_h \stackrel{\text{def}}{=} \{\lceil s \rceil_\ell \mid s \in \mathcal{S}\}$
2. $\tilde{P}_h(\tilde{s}' \mid \tilde{s}, a) \stackrel{\text{def}}{=} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \tilde{s}, a) ds'$
3. $\tilde{s}_0 \stackrel{\text{def}}{=} (\lceil s_0 \rceil_\ell, B)$

and $\tilde{C}$ is the cost criterion defined by replacing $f_{s'}$ with its natural finite equivalent $\tilde{f}$.

We see that discretization results in a small impact to both the value and cost that depend on the continuity parameters.

Lemma 19 (Discretization)**.** *For all $h \in [H+1]$, $\tau_h \in \mathcal{H}_h$, and $\pi \in \Pi^D$, we let $\tilde{\tau}_h$ denote τ_h with each state s_t rounded to $\lceil s_t \rceil_\ell$. Then, we have that $\tilde{V}_h^\pi(\tilde{\tau}_h) \geq V_h^*(\tau_h) - \ell(\lambda_r + \lambda_p) H r_{max}(s_{max} - s_{min})(H - h + 1)$ and $\tilde{C}_h^\pi(\tilde{\tau}_h) \leq C_h^*(\tau_h) + \ell(\lambda_c + \lambda_p) H c_{max}(s_{max} - s_{min})(H - h + 1)$. For almost-sure/anytime constraints, the cost incurs an additional factor of $1/\tilde{p}_{min}$, where $\tilde{p}_{min}$ denotes the smallest non-zero transition probability for $\tilde{M}$.*

Overall, using our previous bicriteria on $\tilde{M}$ yields our approximation results.

Theorem 18 (Continuous Bicriteria)**.** *For any SR-criterion CMDP satisfying Assumption 7 and any* $\epsilon > 0$*, the choice of* $\ell_d \overset{def}{=} \frac{\epsilon/2}{(\lambda_r+\lambda_c+\lambda_p)H\max(c_{max},r_{max})(s_{max}-s_{min})}$ *and approximation* $\ell_a \overset{def}{=} \frac{\epsilon/2}{1+(S+1)H}$ *ensures Algorithm 17*$(\tilde{M})$ *is a* (ϵ,ϵ)*-bicriteria running in time* $O\left(H^{6m+1}\tilde{S}^{4m+2}A\left\|c_{max}-c_{min}\right\|_\infty^{3m}/\epsilon^{3m}\right)$*, where* $\tilde{S} = O((\lambda_r+\lambda_c+\lambda_p)H\max(c_{max},r_{max})(s_{max}-s_{min})^2/\epsilon)$*. This time is polynomial so long as* $|s_{max}-s_{min}| = O(|M|)$*. Moreover, almost-sure/anytime constraints enjoy the same guarantee with an additional factor of* $\tilde{p}_{min}$ *in* $\tilde{S}$*.*

Corollary 7 (Simplified)**.** *For continuous-state SR-criterion CMDPs satisfying Assumption 7, there exist polynomial-time* (ϵ,ϵ)*-bicriteria solutions for expectation constraints, almost-sure constraints, anytime-almost-sure constraints, and any combinations of these constraints.*

6.6 Conclusion

In this chapter, we studied the question of whether polynomial-time approximation algorithms exist for many of the classic formulations studied in the CRL literature. We conclude that for the vast majority of constraints, including all the standard constraints, polynomial-time approximability is possible. We demonstrated this phenomenon by developing polynomial-time bicriteria approximations with the strongest possible guarantees for a general class of constraints that can be written in a form that satisfies general policy evaluation equations. Overall, our work resolves the polynomial-time approximability of many settings, some of which have lacked any polynomial-time algorithm for over a decade. In particular, we are the first to develop a polynomial-time algorithm with any kind of guarantee for chance constraints and non-homogeneous constraints.

Chapter 7

Conclusion

In this work, our aim was to advance the field of safe MARL through the answering of several long-standing open questions and the development of brand new polynomial-time algorithms for fundamental problems. We achieved this goal by studying safety both from other agents through Adversarial RL as well as the environment itself through Constrained RL. A summary of our main results from each chapter follows below.

1. In Chapter 2, we rigorously studied the attack and defense problems of reinforcement learning. We showed that for any attack's surface, a malicious attacker can optimally and efficiently maximize its own rewards by solving a higher lever meta-MDP. When perceived-state attacks are not allowed, we showed that the victim can also compute an optimal defense policy in polynomial time using a robust backward induction algorithm.

2. In Chapter 3, we studied misinformation attacks on two-player MGs. When the victim player only knows a false attacker reward function, we showed how the game plays out under worst-case rationality. Then, we showed how the attacker can compute its worst-case optimal policy in polynomial time. Using this

method as a subroutine, the attacker can exploit the universal assumption of rationality in MARL to compute an optimal dominant-policy inception attack in polynomial time.

3. In Chapter 4, we formalized and rigorously studied anytime-constrained cMDPs. We presented a fixed-parameter tractable reduction based on cost augmentation and safe exploration that yields efficient planning and learning algorithms when the cost precision is $O(\log(|M|))$. In addition, we developed efficient planning and learning algorithms to find ϵ-approximately feasible policies with optimal value whenever the maximum supported cost is $O(\text{poly}(|M|)\max(1, |B|))$.

4. In Chapter 5, we studied the computational complexity of computing deterministic policies for CRL problems. Our main contribution was the design of an FPTAS, Algorithm 13, that solves (CON) for any cMPD and TSR criteria under mild reward assumptions. In particular, our method is an additive-FPTAS if the cMDP's rewards are polynomially bounded, and is a relative-FPTAS if the cMDP's rewards are non-negative. Our work finally resolves the long-standing open questions of polynomial-time approximability for 1) anytime-constrained policies, 2) almost-sure-constrained policies, and 3) deterministic expectation-constrained policies.

5. In Chapter 6, we developed polynomial-time bicriteria approximations with the strongest possible guarantees for a general class of constraints that can be written in a form that satisfies general policy evaluation equations. Overall, our work resolves the polynomial-time approximability of many settings, some of which have lacked any polynomial-time algorithm for over a decade. In particular, we are the first to develop a polynomial-time algorithm with any kind of guarantee for chance constraints and non-homogeneous constraints.

Overall, these results advanced the theoretical foundations of safe MARL and we hope provide new insights that may lead to future practical deployments.

Future Directions. One prevailing theme in all of these works is the use of worst-case analysis. As we have seen, many of these problems are incredibly hard in the worst case, but it is not clear if these problems are as hard in practice. One way to circumvent these worst-case hardness dead-ends is to focus on average case and the more refined smoothed analysis. Such an analysis could give strong evidence that many of these problems are not as hard practice and even suggest useful strategies for solving them. For example, a beyond-worst case analysis for POMGs could enable the efficient computation of robust policies even in the face of observation attacks. Similarly, such an analysis could enable the efficient computation of truly feasible constrained policies even under a complex set of heterogeneous constraints.

On a different note, our constrained works focus on constraints on cumulative costs. However, especially in goal-based RL settings, covering and other more complex constraints are not easily modeled by a cumulative cost. A more applicable formulation of constrained RL allows general set-based constraints. For example, in a disaster relief scenario, an autonomous vehicle must visit each survivor location. Such a constraint cannot be guaranteed even by an almost sure constraint since revisiting the same location multiple times could falsely indicate to the vehicle that all survivors have been rescued. Instead a constraint on the actual set of visited locations is necessary to guarantee survivor safety. The most tractable of these set constraints are submodular constraints. Developing a new framework to handle MDPs and MGs with submodular constraints would be the most realistic and impactful future work.

Appendix A

Chapter 2 Appendix

A.1 Proofs for Optimal Attacks

The proof of the Proposition 1 is immediate from the definition of each attack surface and the well-known fact that any MDP admits an optimal deterministic, Markovian policy. Proposition 2 follows from the complexity results given in [91] and [101].

Proof of Theorem 1.

Proof. The key idea is each meta-state transitions according to either, O, π, R, or P, and since each of these quantities is linear, the meta-transitions are linear. Also, as the rewards are a deterministic projection, they are also linear. If M and π are linear then,

$$\mathcal{O}(o \mid s) = \langle \phi(s), \gamma(o) \rangle \,, \quad \pi(a \mid o) = \langle \psi(o), \delta(a) \rangle \,, \quad R(r \mid s, a) = \langle \phi(s, a), \theta(r) \rangle \,,$$

$$P(s' \mid s, a) = \langle \phi(s, a), \mu(s') \rangle \,.$$

We first design a feature vector $\bar{\phi}(\bar{s}, \bar{a})$ and vector $\bar{\mu}(\bar{s})$ that captures the transitions.

Let $\bar{s} \in \mathcal{S}$, $\bar{a} \in \mathcal{A}(\bar{s})$, and $\bar{s}' \in \mathcal{S}$. From the definition of $\bar{M}$,

1. If $\bar{s} = s$, then $\bar{a} = s^\dagger$ and $\bar{s}' = (s^\dagger, o)$:

$$\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := \mathcal{O}(o \mid s^\dagger) = \left\langle \phi(s^\dagger), \gamma(o) \right\rangle .$$

   Define $\bar{\phi}(s, s^\dagger) = \phi(s^\dagger)$ and $\bar{\mu}(s^\dagger, o) = \gamma(o)$. Then,

$$\left\langle \phi(s^\dagger), \gamma(o) \right\rangle = \left\langle \bar{\phi}(\bar{s}, \bar{a}), \bar{\mu}(\bar{s}') \right\rangle .$$

2. If $\bar{s} = (s, o)$, then $\bar{a} = o^\dagger$ and $\bar{s}' = (s, o^\dagger, a)$:

$$\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := \pi(a \mid o^\dagger) = \left\langle \psi(o^\dagger), \delta(a) \right\rangle .$$

   Define $\bar{\phi}((s, o), o^\dagger) = \psi(o^\dagger)$ and $\bar{\mu}(s, o^\dagger, a) = \delta(a)$. Then,

$$\left\langle \psi(o^\dagger), \delta(a) \right\rangle = \left\langle \bar{\phi}(\bar{s}, \bar{a}), \bar{\mu}(\bar{s}') \right\rangle .$$

3. If $\bar{s} = (s, o, a)$, then $\bar{a} = a^\dagger$ and $\bar{s}' = (s, o, a^\dagger, r)$:

$$\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := R(r \mid s, a^\dagger) = \left\langle \phi(s, a^\dagger), \theta(r) \right\rangle .$$

   Define $\bar{\phi}((s, o, a), a^\dagger) = \phi(s, a^\dagger)$ and $\bar{\mu}(s, o, a, r) = \theta(r)$. Then,

$$\left\langle \phi(s, a^\dagger), \theta(r) \right\rangle = \left\langle \bar{\phi}(\bar{s}, \bar{a}), \bar{\mu}(\bar{s}') \right\rangle .$$

4. If $\bar{s} = (s, o, a, r)$, then $\bar{a} = r^\dagger$ and $\bar{s}' = s'$:

$$\bar{P}(\bar{s}' \mid \bar{s}, \bar{a}) := P(s' \mid s, a) = \langle \phi(s, a), \mu(s') \rangle .$$

Define $\bar{\phi}((s,o,a,r),r^\dagger) = \phi(s,a)$ and $\bar{\mu}(s') = \mu(s')$. Then,

$$\langle \phi(s,a), \mu(s') \rangle = \left\langle \bar{\phi}(\bar{s},\bar{a}), \bar{\mu}(\bar{s}') \right\rangle .$$

We lift each vector to dimension $d = \max(d(\pi), d(M))$ so each vector is in the same dimension.

Now, to capture rewards, we will need to lift the vectors to one dimension higher. We add one entry to each vector. The entry is $g(s,a,r^\dagger)$ for meta-states of form (s,a,r) and meta-action $r^\dagger$, making the new vector $\phi'(\bar{s},\bar{a}) = \begin{bmatrix} g\left(s,a,r^\dagger\right) \\ \bar{\phi}(\bar{s},\bar{a}) \end{bmatrix}$. All other meta-states and meta-actions have a 0 in the new entry, or $\phi'(\bar{s},\bar{a}) = \begin{bmatrix} 0 \\ \bar{\phi}(\bar{s},\bar{a}) \end{bmatrix}$. We define $\bar{\theta} = e_1$, the basis vector with a 1 in the entry that we just added to the features. From the definition of $\bar{M}$,

$$\bar{r}(s,s^\dagger) = \bar{r}((s,a),a^\dagger) = \bar{r}((s,a),s^\dagger) = 0 = \left\langle \bar{\phi}'(\bar{s},\bar{a}), e_1 \right\rangle .$$

$$\bar{r}((s,a,r),r^\dagger) = g(s,a,r^\dagger) = \left\langle \bar{\phi}'(\bar{s},\bar{a}), e_1 \right\rangle .$$

Thus, $\bar{M}$ is linear with dimension at most $1 + \max(d(\pi), d(M))$. □

We note that for the finite-horizon case, the proof remains the same except for one aspect. If the attacker's reward g is time-dependent, then the feature vectors involving $r^\dagger$ must save $g_h(s,a,r^\dagger)$ for all h. Thus the dimension would be $H + 1 + \max(d(\pi), d(M))$ if we modify the proof in the obvious way.

More Details of Remark 2. When the victim's policy is not Markovian, i.e. uses some amount of history, the construction of $\bar{M}$ needs to be slightly modified.

Suppose $\mathcal{M}$ denotes the victim's finite memory. For any $m \in \mathcal{M}$, we let $m(\cdot)$ denote the victim's updated memory upon receiving new information. Since the victim has a finite amount of memory, we assume that when the memory is updated, it removes the oldest saved information to make space for the new information. We also assume here, the attacker is attacking a Markov game G so that π is now a joint policy as described in Remark 3. We define the attacker's meta-MDP as $\bar{M} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{P}, \bar{r}, \gamma, \bar{\mu})$, where,

- $\bar{\mathcal{S}} = (\mathcal{M} \times \mathcal{S}) \cup (\mathcal{M} \times \mathcal{S} \times O) \cup (\mathcal{M} \times \mathcal{S} \times O \times \mathcal{A}) \cup (\mathcal{M} \times \mathcal{S} \times O \times \mathcal{A} \times \mathcal{R})$.
- $\bar{\mathcal{A}} \subseteq \mathcal{S} \cup \mathcal{A} \cup \mathcal{R}$ consists of all the attacker's potential manipulations of the interaction. The meta-action space is meta-state dependent: $\mathcal{A}(m, s) \subseteq \mathcal{S}$, $\mathcal{A}(m, s, o) \subseteq \mathcal{O}$, $\mathcal{A}(m, s, o, a) \subseteq \mathcal{S} \cup \mathcal{A}$, and $\mathcal{A}(m, s, o, a, r) \subseteq \mathcal{R}$.
- Suppose that $m \in \mathcal{M}$, $s \in \mathcal{S}$, $o \in O$, $a \in \mathcal{A}$, and $r \in \mathcal{R}$. Then,
  - If $\bar{a} = s^\dagger$ is a true-state attack at the first subtime, then G's state becomes $s^\dagger$ and generates an observation according to $P(s^\dagger)$.

$$\bar{P}((m, s^\dagger, o) \mid (m, s), s^\dagger) := P(o \mid s^\dagger).$$

  - If $\bar{a} = o^\dagger \in O$ is an observation attack, then the agents choose a joint action $a \sim \pi(m, o^\dagger)$. Also, the agent sees the attacked observation and so updates its memory to $m(o^\dagger)$.

$$\bar{P}((m(o^\dagger), s, o^\dagger, a) \mid (m, s, o), o^\dagger) := \pi(a \mid m, o^\dagger).$$

  - If $\bar{a} = a^\dagger \in \mathcal{A}$ is an action attack, G receives action $a^\dagger$. Thus, G generates reward according to $R(s, a^\dagger)$. The agents also update their memory to

$m(a)$.

$$\bar{P}((m(a), s, o, a^\dagger, r) \mid (m, s, o, a), a^\dagger) := R(r \mid s, a^\dagger).$$

– If $\bar{a} = s^\dagger \in \mathcal{S}$ is a true-state attack at the third subtime, G's state becomes $s^\dagger$. Thus, G generates reward according to $R(s^\dagger, a)$. The agents also update their memory to $m(a)$.

$$\bar{P}((m(a), s^\dagger, o, a, r) \mid (m, s, o, a), s^\dagger) := R(r \mid s^\dagger, a).$$

– If $\bar{a} = r^\dagger \in \mathcal{R}$ is a reward attack, G's transitions are not effected. Thus, G transitions normally according to $P(s, a)$. Also, the agent sees the attacked reward and so updates its memory to $m(r^\dagger)$.

$$\bar{P}((m(r^\dagger), s') \mid (m, s, o, a, r), r^\dagger) := P(s' \mid s, a).$$

All other transitions have probability 0.

- Using the same definitions as above,

  – There is no immediate reward to the attacker for non-reward attacks since the agents have not received a reward yet.

  $$\bar{r}((m, s), s^\dagger) = \bar{r}((m, s, o), o^\dagger) = \bar{r}((m, s, o, a), a^\dagger) = \bar{r}((m, s, o, a), s^\dagger) = 0.$$

  – If $\bar{a} = r^\dagger \in \mathcal{R}$, the agents receive reward $r^\dagger$ and so the attacker receives reward according to its reward function $g(s, a, r^\dagger)$.

  $$\bar{r}(\bar{s}, \bar{a}) = \bar{r}((m, s, o, a, r), r^\dagger) = g(s, a, r^\dagger).$$

- $\bar{\mu}(s) = \mu(\varnothing, s)$ for each $s \in \mathcal{S}$ where $\varnothing$ denotes the empty memory, and $\bar{\mu}(\bar{s}) = 0$ otherwise.

Deterministic Markovian policy and deterministic reward. If the agent's policy is deterministic and Markovian, which is the standard for optimal policies, and M is fully-observable with deterministic rewards, the attacker's meta-MDP can be drastically simplified. We define the attacker's meta-MDP as $\bar{M} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \{\bar{P}_h\}, \{\bar{r}_h\}, H, \bar{\mu})$, where,

- $\bar{\mathcal{S}} = \mathcal{S}$.
- $\bar{\mathcal{A}} = (\mathcal{S} \times \{p\}) \cup \mathcal{A} \cup (\mathcal{S} \times \{t\}) \cup \mathcal{R}$, where (s, p) is a perceived state attack and (s, t) is a true state attack.
- $\bar{P}_h(\bar{s}' \mid \bar{s}, \bar{a})$'s definition varies depending on the choice of $\bar{a}$. Suppose that s is the current state at time h. Then,
  - If $\bar{a} = (s^\dagger, p) \in \mathcal{S} \times \{p\}$ is a perceived-state attack,

$$\bar{P}_h(s' \mid s, (s^\dagger, p)) = P_h(s' \mid s, \pi_h(s^\dagger)).$$

  - If $\bar{a} = a^\dagger \in \mathcal{A}$,

$$\bar{P}_h(s' \mid s, a^\dagger) = P_h(s' \mid s, a^\dagger).$$

  - If $\bar{a} = (s^\dagger, t) \in \mathcal{S} \times \{t\}$ is a true-state attack,

$$\bar{P}_h(s' \mid s, (s^\dagger, t)) = P_h(s' \mid s^\dagger, \pi_h(s^\dagger)).$$

  - If $\bar{a} = r^\dagger \in \mathcal{R}$,

$$\bar{P}_h(s' \mid s, r^\dagger) = P_h(s' \mid s, \pi_h(s)).$$

and all other transitions have probability 0 at time h.

- $\bar{r}_h(\bar{s}, \bar{a})$'s definition similarly depends on $\bar{a}$ as well as the form of $\bar{s}$. Using the same definitions as above, we have

  - If $\bar{a} = (s^\dagger, p) \in \bar{\mathcal{S}} \times \{p\}$ is a perceived-state attack,

    $$\bar{r}_h(s, (s^\dagger, p)) = g(s, r_h(s, \pi_h(s^\dagger))).$$

  - If $\bar{a} = a^\dagger \in \bar{\mathcal{A}}$,

    $$\bar{r}_h(s, a^\dagger) = g(s, r_h(s, a^\dagger)).$$

  - If $\bar{a} = (s^\dagger, t) \in \mathcal{S} \times \{t\}$ is a true-state attack,

    $$\bar{r}_h(s, (s^\dagger, t)) = g(s, r_h(s^\dagger, \pi_h(s^\dagger))).$$

  - If $\bar{a} = r^\dagger \in \mathcal{R}$,

    $$\bar{r}_h(s, r^\dagger) = g(s, r^\dagger).$$

- $\bar{\mu}(s) = \mu(s)$ for each $s \in \mathcal{S}$ and $\bar{\mu}(\bar{s}) = 0$ otherwise.

A.2 Proofs for Optimal Defense

The proof of Proposition 3 and Proposition 4 is immediate from the definitions of the defense problem, the victim-attacker-environment interaction, and the definition of turn-based stochastic games [49].

Proof of Proposition 5.

Proof. We show that the defense problem with observation or perceived-state attacks captures solutions to POMDPs. As such, all hardness results for POMDPs apply to Defense. Namely, computing ϵ-optimal deterministic Markovian policies is NP-hard [73]. Also, for the discounted infinite horizon case, computing an optimal stationary stochastic memory-less policy is NP-hard [111].

We note any POMDP can be formulated as an equivalent POMDP with a deterministic observation function that is only polynomially sized larger. Thus, we can focus on POMDPs with a deterministic observation function $o : \mathcal{S} \to \mathcal{O}$. We can also assume that $\mathcal{O} \subseteq \mathcal{S}$. Given such a POMDP, define M to be the same as the POMDP ignoring the observation part, and define the set constraints for a perceived-state attack to be the singleton $B(s) = \{o(s)\}$. This ensures the only feasible attack ν is exactly the observation function o. Thus, solving the defense problem for a maximum policy π over some class of policies Π is exactly equivalent to solving the POMDP over that class of policies Π. Hardness then follows.

□

The proof of Proposition 6 follows from Observation 1 and the stated complexity results.

Proof of Theorem 2. We present a formal backward induction algorithm for computing defenses. We define $V^*_{h,1}(s) := \max_{\pi \in \Pi_h} \min_{\nu \in BR_h(\pi)} V^{\pi,\nu}_{h,1}(s)$, where Π_h is the set of Markovian partial policies from time h forward, $BR_h(\pi) := \arg\max_{\nu \in N_h} V^{\pi,\nu}_{h,2}(s)$, and $V^{\pi,\nu}_{h,i}(s)$ is the value of the stage game (h, s) for player i (i.e. their expected value from time h onward starting from state s). It is clear that $V^*_{h,1}(s)$ admits optimal substructure: if $\pi^*_h(s)$ is a maximizer of $V^*_{h,1}(s)$, then it must be the case that $\pi^*_{h+1}(s')$ is a maximizer of $V^*_{h+1,1}(s')$ for any s' reachable with non-zero probability under $\pi^*_h(s)$. Otherwise, more value could be achieved by improving the value in the future.

This is the standard argument for why rollback successfully computes a subgame perfect equilibrium in sequential games. Here it is crucial that the defense problem is captured by a turn-based game, which gives us this sequential structure. Given this observation, it is clear that $V^*_{h,i}(s)$ can be computed recursively as follows. Suppose the victim already computed its optimal π^*_{h+1}. Then, $\pi^*_h(s)$ can be computed as a maximizer of $V^*_{h,1}(s)$, where:

$$\begin{aligned} V^*_{h,1}(s) := \max_{a \in \mathcal{A}} \min_{a^\dagger \in BR_h(s,a)} \mathbb{E}_{r \sim R_h(s,a^\dagger)} \Big[\min_{r^\dagger \in BR_h(s,a^\dagger,r)} r^\dagger \\ + \mathbb{E}_{s' \sim P_h(s,a^\dagger)} \left[\min_{s^\dagger \in BR_{h+1}(s')} V^*_{h+1,1}(s^\dagger) \right] \Big]. \end{aligned} \tag{A.1}$$

Each best-response function is defined as follows. In each case, the attacker must wait for further information and then adapt to each realization. Hence the separate best-response functions.

$$BR_h(s,a) := \arg\max_{a^\dagger \in \bar{\mathcal{A}}_h(s,a)} \mathbb{E}_{r \sim R_h(s,a^\dagger)} V^*_{h,2}(s,a,r). \tag{A.2}$$

$$BR_h(s,a,r) := \arg\max_{r^\dagger \in \bar{\mathcal{A}}_h(s,a,r)} g_h(s,a,r^\dagger) + \mathbb{E}_{s' \in P_h(s,a)} V^*_{h+1}(s'). \tag{A.3}$$

$$BR_{h+1}(s) := \max_{s^\dagger \in \bar{\mathcal{A}}_{h+1}(s)} V^*_{h+1,2}(s^\dagger, \pi_{h+1}(s^\dagger))) \tag{A.4}$$

In all cases, $V^*_{h+1,2}(\cdot)$ is defined to be the maximizing value for each corresponding BR set.

In the zero-sum case, we get the even simpler expression,

$$V^*_{h,1}(s) = \max_{a \in \mathcal{A}} \min_{a^\dagger \in \bar{\mathcal{A}}_h(s,a^\dagger)} \mathbb{E}_{r \sim R_h(s,a^\dagger)} \left[\min_{r^\dagger \in \bar{\mathcal{A}}(s,a^\dagger,r)} r^\dagger + \mathbb{E}_{s' \sim P_h(s,a^\dagger)} \left[\min_{s^\dagger \in \bar{\mathcal{A}}(s')} V^*_{h+1,1}(s^\dagger) \right] \right]. \tag{A.5}$$

Since turn-based games admit deterministic solutions, it is clear that the victim can compute its optimal defense in polynomial time by simply brute-force computing each max and min in the above expressions, which are finite optimizations in the tabular case.

A.3 Code Details

We conducted our experiments using standard python3 libraries. We provide our code in a jupyter notebook with the example grid being hard-coded in to ensure the experiments can be easily reproduced[1].

[1]Code can be found at https://github.com/jermcmahan/Attack-Defense.

Appendix B

Chapter 3 Appendix

B.1 Extended Preliminaries

Normal-form Games. In a (finite) normal-form game, two players compete simultaneously to maximize their reward. Suppose the first player, the victim, has n pure strategies and the second player, the attacker, has m pure strategies. Let $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times m}$ denote the reward matrices for the victim and attacker, respectively. We may represent a pure strategy by a one-hot vector, so $e_i \in \mathbb{R}^n$ corresponds to the victim's strategy i and $e_j \in \mathbb{R}^m$ the attacker's strategy j. Let $\Delta(k) := \left\{ s \in [0,1]^k \mid \sum_{i=1}^k s_i = 1 \right\}$ denote the set of mixed strategies, where choosing $s \in \Delta(k)$ corresponds to playing e_i with probability s_i. For a pair of mixed strategies $x \in \Delta(n)$ and $y \in \Delta(m)$, the expected rewards to the victim and attacker are $x^\top A y$ and $x^\top B y$, respectively.

Nash Equilibrium. Solutions to games manifest as equilibrium concepts, among which the most famous is the *Nash Equilibrium* (NE) [83]. An NE of a bimatrix

game is a pair of strategies $(x^*, y^*) \in \Delta(n) \times \Delta(m)$ satisfying,

$$x^* \in \arg\max_{x \in \Delta(n)} x^\top A y^* \quad \text{and} \quad y^* \in \arg\max_{y \in \Delta(m)} x^{*\top} B y.$$

In words, x^* and y^* are mutual best-responses to each other. We let $NE(A, B)$ denote the set of all NEs for the game (A, B).

Security Strategies. Another solution concept is a *maximin strategy* or *security strategy*, which is a pair (x^*, y^*) given by,

$$x^* \in \arg\max_{x \in \Delta(n)} \min_{y \in \Delta(m)} x^\top A y \quad \text{and} \quad y^* \in \arg\max_{y \in \Delta(m)} \min_{x \in \Delta(n)} x^\top B y. \tag{B.1}$$

In a *zero-sum* game ($B = -A$), the Minimax Theorem [112] implies (x^*, y^*) is a NE if and only if it is a maximin strategy pair. Note that a game may have multiple NEs and maximin strategies. However, in zero-sum games, each player receives the same expected reward in every NE, which we denote by p_v^{NE} and p_e^{NE} respectively.

Markov Game Solutions. Equilibrium concepts can be defined for a Markov Game by viewing it as a (very large) bimatrix game with reward matrices $(V_1^{\pi_1,\pi_2})_{\pi_1,\pi_2}$ and $(V_2^{\pi_1,\pi_2})_{\pi_1,\pi_2}$. To avoid this complexity blowup, many works focus on *Markov Perfect Equilibrum* (MPE), which requires the stricter property that a policy pair is an equilibrium at *every* stage game, not just at stage $h = 1$. Formally, (π_1^*, π_2^*) is a MPE if, for all $(h, s) \in [H] \times S$,

$$V_{1,h}^{\pi_1^*,\pi_2^*}(s) = \max_{\pi_1 \in \Pi_1} V_{1,h}^{\pi_1,\pi_2^*}(s) \quad \text{and} \quad V_{2,h}^{\pi_1^*,\pi_2^*}(s) = \max_{\pi_2 \in \Pi_2} V_{2,h}^{\pi_1^*,\pi_2}(s).$$

B.2 Proofs for Section 3.2

All the proofs from section 2 are immediate from the arguments given in the main text.

B.3 Proofs for Section 3.3.1

As mentioned in the main text, the proof of Lemma 1 is immediate from standard bimatrix game theory [28].

B.3.1 Proof of Lemma 1

The proof is immediate from the argument given in the main text.

B.3.2 Proof of Lemma 2

To construct the dual in Figure 3.3, we introduce a dual vector $w \in \mathbb{R}^K_{\geq 0}$ corresponding to the inequality constraints and a dual variable $v \in \mathbb{R}$ corresponding to the equality constraint. We multiply these dual variables by their respective constraints and add them to the objective to get the equivalent optimization:

$$\max_{w \geq 0, v} \min_{x \geq 0} x^\top B y + (z^* 1^\top - x^\top A')w + (x^\top 1 - 1)v$$

By rearranging the objective to be in terms of x, we get:

$$\max_{w \geq 0, v} \min_{x \geq 0} x^\top (By - A'w + 1v) + z^* 1^\top w - v$$

Moving the terms involving x into the constraints then gives the Dual:

$$\begin{aligned} \max_{w \geq 0, \alpha} \quad & z^* 1^\top w - \alpha \\ \text{s.t.} \quad & \alpha + e_i^\top B y - e_i^\top A' w \geq 0 \quad \forall i \in [n], \end{aligned}$$

Applying $\max_{y \in \Delta(m)}$ outside of the Dual, yields the attacker's LP 3.2b:

$$\begin{aligned} \max_{y, w \in \mathbb{R}^K, \alpha \in \mathbb{R}} \quad & z^* 1^\top w - \alpha \\ \text{s.t.} \quad & \alpha + e_i^\top B y - e_i^\top A' w \geq 0 \quad \forall i \in [n] \\ & 1^\top y = 1, \quad y \geq 0 \quad w \geq 0. \end{aligned}$$

The fact that there exist optimal solutions, i.e., $\Pi_2^*(R_2^\dagger) \neq \varnothing$, follows from LP 3.2b being feasible and bounded. Specifically, it is easily seen that choosing $y = e_1$, $w = 0$, and $\alpha = \max_{i \in [n]} |e_i^\top B e_1$ gives a feasible solution to LP 3.2b. Boundedness follows from the fact that by LP duality, LP 3.2b is value equivalent to the original problem $\max_{y \in \Delta(m)} \min_{x \in \Pi_1^*(R_2^\dagger)} x^\top B y$, which is bounded being that (A, B) is a finite normal-form game. This completes the proof.

B.3.3 Proof of Theorem 3

The proof is immediate from Lemma 2.

B.3.4 Proof of Lemma 3

From Theorem 3 and the definition of Q^*, it suffices to show that V^* satisfies the following optimality equations:

$$V^*_{1,h}(s) = \max_{\pi_{1,h}(s)\in\Delta(n)} \min_{\pi_{2,h}(s)\in\pi_{1,h}(s)} \mathbb{E}_{a\sim\pi_{1,h}(s)} \left[R_{1,h}(s,a) + \sum_{s'} P_h(s' \mid s,a) V^*_{1,h+1}(s') \right], \tag{B.2}$$

and,

$$V^*_{2,h}(s) = \max_{\pi_{2,h}(s)\in\Delta(m)} \min_{\pi_{1,h}(s)\in\Pi^*_{1,h}(s)} \mathbb{E}_{a\sim\pi_{1,h}(s)} \left[R_{2,h}(s,a) + \sum_{s'} P_h(s' \mid s,a) V^*_{2,h+1}(s') \right], \tag{B.3}$$

where $\Pi^*_{1,h}(s)$ is the set of maximizers to (B.2). This follows from similar arguments to the proof of the NashVI algorithm [62] but with an added constraint set. For completeness, we give a full proof.

Proof. We show (B.3). The proof of (B.2) follows even easier as the constraint set is fixed in advance, independent of the attacker's actions. We proceed by induction on h. For the base case, consider the final time step $h = H + 1$. The claim is trivial as both values are 0. For the inductive step, consider any time step $h < H$ and fix any $s \in S$. Applying the bellman-consistency equations to the definition of $V^*_{2,h}(s)$ yields:

$$V^*_{2,h}(s) = \max_{\pi_2\in\Pi_2} \min_{\pi_1\in\Pi^*_1(R^\dagger_2)} \mathbb{E}_{a\sim\pi_{1,h}(s)} \left[R_{2,h}(s,a) + \sum_{s'} P_h(s' \mid s,a) V^\pi_{2,h}(s') \right].$$

Observe that the expression decomposes: the expectation only considers the policies at the current state and time, $(\pi_{1,h}(s), \pi_{2,h}(s))$, and the summation only considers the policies at future time steps. Consequently, we can break down the $\max_{\pi_2\in\Pi_2}$ into the separate optimizations: $\max_{\pi_{2,h}(s)\in\Delta(m)}$ and $\max_{\pi_2\in\Pi_{2,h+1}(s')}$ for each $s' \in S$, where $\Pi_{2,h+1}(s')$ is the set of partial policies for the attacker from time $h+1$ onwards

starting at state s'.

Similarly, we can break down the $\min_{\pi_1 \in \Pi_1^*(R_2^\dagger)}$ into the separate optimizations: $\min_{\pi_{1,h}(s) \in \Pi_{1,h}^*(s)}$ and $\min_{\pi_1 \in \Pi_{1,h}^*(s')}$ for each $s' \in S$. This yields the equivalent optimization:

$$\max_{\pi_{2,h}(s) \in \Delta(m)} \max_{\pi_2 \in \times_{s'} \Pi_{2,h+1}(s')} \min_{\pi_{1,h}(s) \in \hat{\Pi}_{1,h}^*(s)} \min_{\pi \in \times_{s'} \Pi_{1,h}^*(s')} \mathbb{E}_{\pi_{1,h}(s), \pi_{2,h}(s)} \left[\ldots \right].$$

Now, consider the summation term inside of the optimization:

$$\mathbb{E}_{\pi_{1,h}(s), \pi_{2,h}(s)} \left[\sum_{s'} P_h(s' \mid s, a) V_{2,h+1}^{\pi}(s') \right].$$

We can apply linearity of expectation to get the equivalent term:

$$\sum_{s'} \mathbb{E}_{\pi_{1,h}(s), \pi_{2,h}(s)} \left[P_h(s' \mid s, a) V_{2,h+1}^{\pi}(s') \right].$$

Also, since $V_{2,h+1}^{\pi}(s')$ depends only on the partial policies at future steps, $V_{2,h+1}^{\pi}(s')$ is constant with respect to $(\pi_{1,h}(s), \pi_{2,h}(s))$ so can be pulled out of the summation to get the equivalent term:

$$\sum_{s'} V_{2,h+1}^{\pi}(s') \mathbb{E}_{\pi_{1,h}(s), \pi_{2,h}(s)} \left[P_h(s' \mid s, a) \right].$$

Now, by the induction hypothesis, we know for any s' at time $h+1$,

$$V_{2,h+1}^*(s') = \max_{\pi_{2,h+1}(s') \in \Pi_{2,h+1}(s')} \min_{\pi_{1,h+1}(s') \in \Pi_{1,h+1}^*(s')} \mathbb{E}_{\pi_{1,h+1}(s'), \pi_{2,h+1}(s')} \left[R_{2,h+1}(s', a) + \sum_{s'} P_{h+1}(s'' \mid s', a) V_{2,h+2}^*(s'') \right].$$

Since the term $V_{2,h+2}^*(s'')$ is fixed and shared amongst all s' at time $h+1$, we see the

only variation in the stage value $V^*_{2,h+1}(s')$ comes from choosing $(\pi_{1,h+1}(s'), \pi_{2,h+1}(s'))$ (i.e. varying the future partial policy cannot increase the objective value). These can be independently chosen for all s' at time $h+1$. Thus, the optimization problems $\max_{\pi_2 \in \Pi_{2,h+1}(s')} \min_{\pi_1 \in \Pi^*_{1,h+1}(s')} V^{\pi}_{2,h+1}(s') = V^*_{2,h+1}(s')$ are separable over s'. Thus, we can bring the maximin over partial policies into the summation to get the term:

$$\sum_{s'} \max_{\nu \in \Pi_{2,h+1}(s')} \min_{\pi \in \Pi^*_{1,h+1}(s')} V^{\pi}_{2,h+1}(s') \mathbb{E}_{\pi_{1,h}(s), \pi_{2,h}(s)} \left[P_h(s' \mid s, a) \right].$$

Since $V^*_{2,h+1}(s') = \max_{\pi_2 \in \Pi_{2,h+1}(s')} \min_{\pi_1 \in \Pi^*_{1,h+1}(s')} V^*_{2,h+1}(s')$, the expression becomes:

$$\sum_{s'} V^*_{2,h+1}(s') \mathbb{E}_{\pi_{1,h}(s), \pi_{2,h}(s)} \left[P_h(s' \mid s, a) \right].$$

As $V^*_{2,h+1}(s')$ is still constant with respect to $(\pi_{1,h}(s), \pi_{2,h}(s))$, we can reverse the previous steps of pulling out this term and applying linearity of expectation to get the final expression:

$$V^*_{2,h}(s) = \max_{\pi_{2,h}(s) \in \Delta(m)} \min_{\pi_{1,h}(s) \in \hat{\Pi}^*_{1,h}(s)} \mathbb{E}_{\pi_{1,h}(s), \pi_{2,h}(s)} \left[R_{2,h}(s,a) + \sum_{s'} P_h(s' \mid s, a) V^*_{2,h+1}(s') \right]$$

□

B.3.5 Proof of Theorem 4

The proof is immediate from Lemma 3.

B.4 Proofs for Section 3.3.2

B.4.1 Proof of Lemma 4

The proof is immediate from the argument given in the main text.

B.4.2 Proof of Lemma 5

The proof follows similarly to the proof of Lemma 3 and the arguments from the main text.

B.4.3 Proof of Theorem 5

The proof is immediate from Lemma 5.

Appendix C

Chapter 4 Appendix

C.1 Proofs for Section 4.2

C.1.1 Proof of Proposition 8

Proof. Fix any $H \geq 2$. For any $h \in [H-1]$, let π be a cost-history-dependent policy that does not record the cost at time h. For any such π, we construct a cMDP instance for which π is arbitrarily suboptimal. This shows that any class of policies that does not consider the full cost history is insufficient to solve (ANY). In particular, the class of Markovian policies does not suffice.

Consider the simple cMDP M_h defined by a single state, $\mathcal{S} = \{0\}$, two actions, $\mathcal{A} = \{0, 1\}$, and horizon H. The initial state is trivially 0 and the transitions are trivially self-loops from 0 to 0. Importantly, M_h has non-stationary rewards and costs. The rewards are deterministic. For any $t \neq h+1$, $r_t(s, a) = 0$. For some large $x > 0$, $r_{h+1}(s, 1) = x$ and $r_{h+1}(s, 0) = 0$. The costs are deterministic except at time h. For any $t \notin \{h, h+1\}$, $c_t(s, a) = 0$. For $t = h+1$, $c_{h+1}(s, 1) = B$ and

$c_{h+1}(s, 0) = 0$. For $t = h$, the costs are random:

$$C_h(s, a) := \begin{cases} B & \text{w.p. } \frac{1}{2} \\ 0 & \text{w.p. } \frac{1}{2} \end{cases}$$

The budget is any $B > 0$.

Clearly, an optimal cost-history-dependent policy can choose any action it likes other than at time $h + 1$. At time $h + 1$, an optimal policy chooses $a = 1$ if the cost incurred at time h was 0 and otherwise chooses action $a = 0$. The value of the optimal policy is $x/2$ since the agent receives total reward x whenever $c_h = 0$, which is half the time, and otherwise receives total reward 0. Thus, $V^*_{M_h} = x/2$.

On the other hand, consider π's performance. Since π does not record c_h, it cannot use c_h to make decisions. Hence, $p := \mathbb{P}^\pi_{M_h}[a_{h+1} = 1]$ is independent of c_h's value. If $p > 0$ then with probability $1/2p > 0$, the agent accrues cost B at both time h and time $h + 1$ so violates the constraint. Thus, if π is feasible, it must satisfy $p = 0$. Consequently, π can never choose $a = 1$ at time $h + 1$ and so can never receive rewards other than 0. Thus, $V^\pi_{M_h} = 0 << x/2 = V^*_{M_h}$. By choosing x large enough, we see policies that do not consider the entire cost history can be arbitrarily suboptimal. By applying this argument to $H = 2$ and $h = 1$, we see that Markovian policies can be arbitrarily suboptimal and so do not suffice to solve anytime-constrained cMDPs. □

C.1.2 Proof of Corollary 1

Proof. Since optimal policies for expectation-constrained cMDPs are always Markovian, Proposition 8 immediately implies such policies are infeasible or arbitrarily suboptimal. In fact, we can see this using the same construction of M_h.

1. Under an expectation constraint, the optimal policy π can choose $p = 1/2$ and still maintain that $\mathbb{E}_M^\pi[\sum_{t=1}^H c_t] = B/2 + pB = B \leq B$. Thus, such a policy violates the anytime constraint by accumulating cost $2B$ with probability $1/4$. In fact, if we generalize the construction of M_h to have $c_h = \frac{B}{2\delta}$ with probability $\delta > 0$ (where $\delta = 1/2$ in the original construction), then the optimal expectation-constrained policy is the same π but now accumulates cost $\frac{B}{2\delta} + B$ with probability $\delta/2 > 0$. Since δ can be chosen to be arbitrarily small, the violation of the anytime constraint, which is $B/2\delta$, can be arbitrarily large. Even if we relax the policy to just be ϵ-optimal, for any $\epsilon > 0$ we can choose x large enough to where all ϵ-optimal policies still select action 1 with non-zero probability.

2. A similar construction immediately shows the arbitrarily infeasibility of optimal chance-constrained policies. Consider a chance constraint that requires $\mathbb{P}_M^\pi[\sum_{t=1}^H c_t > B] \leq \delta$ for some $\delta > 0$. We can use the same construction as above but with an arbitrarily larger cost of $c_h = y\frac{B}{2\delta}$ for some $y > 0$. Then, an optimal chance constrained policy can always let $p = 1$ since the cost only exceeds budget when $c_h > 0$ which happens with probability δ. Such a policy clearly violates the anytime constraint by $y\frac{B}{2\delta}$, which is arbitrarily large by choosing y to be arbitrarily large. Also, observe this does not require us to consider Markovian policies since whether the budget was already violated at time h or not, the policy is still incentivized to choose action 1 at time $h + 1$ as additional violation does not effect a chance-constraint. Again, considering an ϵ-optimal policy does not change the result.

3.

Suppose instead we computed an optimal policy using a smaller budget B'.

1. For expectation-constraints, to ensure the resultant policy is feasible for anytime constraints, we need that $p = 0$ as before. By inspection, it must be that $B' = B/2$ but then the value of the policy is 0 which is arbitrarily suboptimal as we saw before.

2. For chance-constraints, the situation is even worse. Consider the M_h but with $c_h = B$ w.p. δ. Then, no matter what B' we choose, the resultant policy is not feasible. Specifically, an optimal cost-history-dependent policy under the event that $c_h = B/2$ will then choose $a_{h+1} = 1$ almost surely since the extent of the violation does not matter. But even ignoring this issue, under the event that $c_h = 0$ the policy would then have to choose $a_{h+1} = 0$ which is again arbitrarily suboptimal.

For the knapsack-constrained frameworks, the policy is allowed to violate the budget arbitrarily once per episode. Thus, no matter how we change the budget feasibility is never guaranteed: it will always choose $a_{h+1} = 1$ in any realization. The other frameworks also fail using slight modifications of the constructions above.

□

C.1.3 Proof of Proposition 9

Proof. For continuity with respect to rewards, notice that if a certain reward is not involved in the optimal solution, then any perturbation does not change V^*. On the other hand, if a reward is in an optimal solution, since V^* is defined by an expectation of the rewards, it is clear that V^* is continuous in that reward: a slight perturbation in the reward leads to the same or an even smaller perturbation in V^* due to the probability weighting.

On the other hand, V^* can be highly discontinuous in c and B. Suppose a cMDP

has only one state and two actions with cost 0 and B, and reward 0 and $x \in \mathbb{R}_{>0}$ respectively. Then slightly increasing the cost or slightly decreasing the budget to create a new instance M_ϵ moves a solution value of x all the way down to a solution value of 0. In particular, we see $V^*_M = x >> V^*_{M_\epsilon}$ if we only perturb the budget slightly by some $\epsilon > 0$. □

C.1.4 Proof of Lemma 6

We first formally define the anytime cost of a policy π as,

$$C^\pi := \max_{h \in [H]} \max_{\substack{\tau_h \in \mathcal{H}_h, \\ \mathbb{P}^\pi[\tau_h] > 0}} \bar{c}_h.$$

In words, C^π is the largest cost the agent ever accumulates at any time under any history.

Proof. Consider the deterministic policy π' defined by,

$$\pi'_h(\tau_h) := \max_{\substack{a \in \mathcal{A}, \\ \pi_h(a|\tau_h) > 0}} r_h(s,a) + \mathbb{E}_{c,s'}\left[V^\pi_{h+1}(\tau_h, a, c, s')\right],$$

for every $h \in [H]$ and every $\tau_h \in \mathcal{H}_h$.

We first show that for any $\tau_h \in \mathcal{H}_h$, if $\mathbb{P}^{\pi'}[\tau_h] > 0$ then $\mathbb{P}^\pi[\tau_h] > 0$. This means that the set of partial histories induced by π' with non-zero probability are a subset of those induced by π. Hence,

$$C^{\pi'} = \max_{h \in [H]} \max_{\substack{\tau_h \in \mathcal{H}_h, \\ \mathbb{P}^{\pi'}[\tau_h] > 0}} \bar{c}_h \leq \max_{h \in [H]} \max_{\substack{\tau_h \in \mathcal{H}_h, \\ \mathbb{P}^{\pi}[\tau_h] > 0}} \bar{c}_h = C^\pi.$$

We show the claim using induction on h. For the base case, we consider $h = 1$. By definition, we know that for both policies, $\mathbb{P}^{\pi'}[s_0] = \mathbb{P}^\pi[s_0] = 1$. For the inductive

step, consider any $h \geq 1$ and suppose that $\mathbb{P}^{\pi'}[\tau_{h+1}] > 0$. Decompose τ_{h+1} into $\tau_{h+1} = (\tau_h, a, c, s')$ and let $s = s_h$. As we have shown many times before,

$$0 < \mathbb{P}^{\pi'}[\tau_{h+1}] = \pi'_h(a \mid \tau_h) C_h(c \mid s, a) P_h(s' \mid s, a) \mathbb{P}^{\pi'}[\tau_h]$$

Thus, it must be the case that $\pi'_h(\tau_h) = a$ (since π' is deterministic), $C_h(c \mid s, a) > 0$, $P_h(s' \mid s, a) > 0$, and $\mathbb{P}^{\pi'}[\tau_h] > 0$. By the induction hypothesis, we then know that $\mathbb{P}^{\pi}[\tau_h] > 0$. Since by definition $\pi'_h(\tau_h) = a \in \{a' \in \mathcal{A} \mid \pi_h(a' \mid \tau_h) > 0\}$, we then see that,

$$\mathbb{P}^{\pi}[\tau_{h+1}] = \pi_h(a \mid \tau_h) C_h(c \mid s, a) P_h(s' \mid s, a) \mathbb{P}^{\pi}[\tau_h] > 0$$

This completes the induction.

Next, we show that for any $h \in [H]$ and $\tau_h \in \mathcal{H}_h$, $V_h^{\pi'}(\tau_h) \geq V_h^{\pi}(\tau_h)$. This implies that $V_M^{\pi'} = V_1^{\pi'}(s_0) \geq V_1^{\pi}(s_0) = V_M^{\pi}$ which proves the second claim. We proceed by backward induction on h. For the base case, we consider $h = H + 1$. By definition, both policies achieve value $V_{H+1}^{\pi'}(\tau) = 0 = V_{H+1}^{\pi}(\tau)$. For the inductive step, consider $h \leq H$. By (PE),

$$\begin{aligned}
V_h^{\pi'}(\tau_h) &= r_h(s, \pi'(\tau_h)) + \mathbb{E}_{c,s'}[V_{h+1}^{\pi'}(\tau_{h+1})] \\
&\geq r_h(s, \pi'(\tau_h)) + \mathbb{E}_{c,s'}[V_{h+1}^{\pi}(\tau_{h+1})] \\
&\geq \mathbb{E}_a[r_h(s, a) + \mathbb{E}_{\tilde{c},s'}[V_{h+1}^{\pi}(\tau_{h+1})]] \\
&= V_h^{\pi}(\tau_h).
\end{aligned}$$

The second line used the induction hypotheses. The third lines used the fact that the maximum value is at least any weighted average. This completes the induction.

Thus, we see that π' satisfies $C_M^{\pi'} \leq C_M^{\pi}$ and $V_M^{\pi'} \geq V_M^{\pi}$ as was to be shown.

Furthermore, we see that π' can be computed from π in linear time in the size of π by just computing $V_h^\pi(\tau_h)$ by backward induction and then directly computing a solution for each partial history.

□

C.1.5 Proof of Theorem 6

Proof. We present a poly-time reduction from the knapsack problem. Suppose we are given n items each with a non-negative integer value v_i and weight w_i. Let B denote the budget. We construct an MDP M with $\mathcal{S} = \{0\}$, $\mathcal{A} = \{0, 1\}$, and $H = n$. Naturally, having a single state implies the initial state is $s_0 = 0$, and the transition is just a self-loop: $P(0 \mid 0, a) = 1$ for any $a \in \mathcal{A}$. The rewards of M correspond to the knapsack values: $r_i(s, 1) = v_i$. The costs of M correspond to the knapsack weights: $c_i(s, 1) = w_i$. The budget remains B.

Clearly, any (possibly non-stationary) deterministic policy corresponds to a choice of items for the knapsack. By definition of the rewards and costs, $\pi_h(\cdot) = 1$ if and only if the agent gets reward v_h and accrues cost c_h. Thus, there exists a deterministic $\pi \in \Pi$ with $V_M^\pi \geq V$ and $C_M^\pi \leq B$ if and only if $\exists I \subseteq [n]$ with $\sum_{i \in I} v_i \geq V$ and $\sum_{i \in I} w_i \leq B$. From Lemma 6 if there exists a stochastic optimal policy for the cMDP with value at least V and anytime cost at most B, then there exists a deterministic policy with value at least V and anytime cost at most B. As M can be constructed in linear time from the knapsack instance, the reduction is complete. □

C.1.6 Proof of Theorem 7

Proof. We show that computing a feasible policies for anytime constrained cMDPs with only $d = 2$ constraints is NP-hard via a reduction from Partition. Suppose $X = \{x_1, \ldots, x_n\}$ is a set of non-negative integers. Let $Sum(X) := \sum_{i=1}^n x_i$. We

define a simple cMDP similar to the one in the proof of Theorem 6. Again, we define $\mathcal{S} = \{0\}$, $\mathcal{A} = \{0, 1\}$, and $H = n$. The cost function is deterministic, defined by $c_{i,h}(s, i) = x_h$ and $c_{i,h}(s, 1 - i) = 0$. The budgets are $B_0 = B_1 = Sum(X)/2$.

Intuitively, at time h, choosing action $a_h = 0$ corresponds to placing x_h in the left side of the partition and $a_h = 1$ corresponds to placing x_h in the right side. The total cumulative cost for each constraint corresponds to the sum of elements in each side of the partition. If both sides sum to at most $Sum(X)/2$ then it must be the case that both are exactly $Sum(X)/2$ and so we have found a solution to the Partition problem.

Formally, we show that $\exists \pi \in \Pi_M$ if and only if $\exists Y \subseteq [n]$ with $Sum(Y) = Sum(Z) = Sum(X)/2$ where $Z = X \setminus Y$.

- ($\implies$) Suppose π is a feasible deterministic policy for M (We can assume deterministic again by Lemma 6). Define $Y := \{i \mid \pi_h(s) = 0\}$ and $Z := \{i \mid \pi_h(s) = 1\}$. Since π is deterministic we know that each item is assigned to one set or the other and so $Y \cup Z = X$.

  By definition of the constraints, we have that $\mathbb{P}^{\pi}_{M}[\sum_{h=1}^{H} c_{i,h} \leq B_i] = 1$. Since all quantities are deterministic this means that $\sum_{h=1}^{H} c_{i,h} \leq B_i$. By definition of Y and Z we further see that $Sum(Y) = \sum_{h=1}^{H} c_{0,h} \leq Sum(X)/2$ and $Sum(Z) = \sum_{h=1}^{H} c_{1,h} \leq Sum(X)/2$. Since,

$$\begin{aligned} Sum(X) = Sum(Y \cup Z) &= Sum(Y) + Sum(Z) \\ &\leq Sum(X)/2 + Sum(X)/2 = Sum(X), \end{aligned}$$

  the inequality must be an equality. Using $Sum(Y) = Sum(X) - Sum(Z)$, then implies that $Sum(Y) = Sum(Z) = Sum(X)/2$ and so (Y, Z) is a solution to the partition problem.

Algorithm 18 Compute $\{\bar{\mathcal{S}}_h\}_h$

Input: (M, C, B)

$\bar{\mathcal{S}}_1 = \{(s_0, 0)\}$
for $h \leftarrow 1$ to $H-1$ **do**
  $\bar{\mathcal{S}}_{h+1} = \varnothing$
  for $(s, \bar{c}) \in \bar{\mathcal{S}}_h$ **do**
    for $s' \in \mathcal{S}$ **do**
      for $a \in \mathcal{A}$ **do**
        if $P_h(s' \mid s, a) > 0$ and $\Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1$ **then**
          for $c \in C_h(s, a)$ **do**
            $\bar{\mathcal{S}}_{h+1} \leftarrow \bar{\mathcal{S}}_{h+1} \cup \{(s', \bar{c} + c)\}$
return $\{\bar{\mathcal{S}}_h\}_h$.

- ($\Longleftarrow$) On the other hand, suppose that (Y, Z) is a solution to the partition problem. We can define $\pi_h(s) = 0$ if $h \in Y$ and $\pi_h(s) = 1$ if $h \in Z$. By definition, we see that $Sum(Y) = \sum_{h=1}^{H} c_{0,h} = Sum(X)/2 = B_0$ and $Sum(Z) = \sum_{h=1}^{H} c_{1,h} = Sum(X)/2 = B_1$. Thus, π is feasible for M.

As the construction of M can clearly be done in linear time by copying the costs and computing $Sum(X)/2$, the reduction is polynomial time. Thus, it is NP-hard to compute a feasible policy for an anytime-constrained cMDP.

Since the feasibility problem is NP-hard, it is easy to see approximating the problem is NP-hard by simply defining a reward of 1 at the last time step. Then, if an approximation algorithm yields any finite-value policy, we know it must be feasible since infeasible policies yield $-\infty$ value. Thus, any non-trivial approximately-optimal policy to an anytime-constrained cMDP is NP-hard to compute. □

C.2 Proofs for Section 4.3

The complete forward induction algorithm that computes $\bar{\mathcal{S}}$ as defined in Definition 7 is given in Algorithm 18.

Suppose $\tau_{h+1} \in \mathcal{H}_{h+1}$ is any partial history satisfying $\mathbb{P}^{\pi}_{\tau_h}[\tau_{h+1}] > 0$. By the Markov property (Equation (2.1.11) from [91]), we have that

$$\mathbb{P}^{\pi}_{\tau_h}[\tau_{h+1}] = \pi_h(a \mid \tau_h) C_h(c \mid s, a) P_h(s' \mid s, a). \tag{MP}$$

Thus, it must be the case that $\tau_{h+1} = (\tau_h, a, c, s')$ where $\pi_h(a \mid \tau_h) > 0$, $C_h(c \mid s, a) > 0$, and $P_h(s' \mid s, a) > 0$.

C.2.1 Proof of Lemma 7

We show an alternative characterization of the safe exploration state set:

$$\begin{aligned} SE_h := \Big\{ (s, \bar{c}) \mid \exists \pi \; \exists \tau_h \in \mathcal{H}_h, \mathbb{P}^{\pi}_{\tau_h}[s_h = s, \bar{c}_h = \bar{c}] = 1 \text{ and} \\ \forall k \in [h-1] \; \mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \leq B] = 1 \Big\}. \end{aligned} \tag{C.1}$$

Observe that $\mathbb{P}^{\pi}_{\tau_h}[s_h = s, \bar{c}_h = \bar{c}] = 1$ is equivalent to requiring that for τ_h, $s_h = s$, $\bar{c}_h = \bar{c}$, and $\mathbb{P}^{\pi}[\tau_h] > 0$.

Lemma 20. *For all $h \in [H+1]$, $\bar{\mathcal{S}}_h = SE_h$.*

We break the proof into two claims.

Claim 1. *For all $h \in [H+1]$, $\bar{\mathcal{S}}_h \subseteq SE_h$.*

Proof. We proceed by induction on h. For the base case, we consider $h = 1$. By definition, $\bar{\mathcal{S}}_1 = \{(s_0, 0)\}$. For $\tau_1 = s_0$, we have $\bar{c}_1 = 0$ is an empty sum. Thus, for any $\pi \in \Pi$, $\mathbb{P}^{\pi}[s_1 = s_0, \bar{c}_1 = 0 \mid \tau_1] = 1$. Also, $[h-1] = [0] = \varnothing$ and so the second condition vacuously holds. Hence, $(s_0, 0) \in SE_1$ implying that $\bar{\mathcal{S}}_1 \subseteq SE_1$.

For the inductive step, we consider any $h \geq 1$. Let $(s', \bar{c}') \in \bar{\mathcal{S}}_{h+1}$. By definition,

we know that there exists some $(s, \bar{c}) \in \bar{\mathcal{S}}_h$, $a \in \mathcal{A}$, and $c \in \mathbb{R}$ satisfying,

$$C_h(c \mid s, a) > 0, \ \Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1, \ \bar{c}' = \bar{c} + c, \text{ and } P_h(s' \mid s, a) > 0.$$

By the induction hypothesis, $(s, \bar{c}) \in SE_h$ and so there also exists some $\pi \in \Pi$ and some $\tau_h \in \mathcal{H}_h$ satisfying,

$$\mathbb{P}^{\pi}_{\tau_h}[s_h = s, \bar{c}_h = \bar{c}] = 1 \text{ and } \forall k \in [h-1], \ \mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \leq B] = 1.$$

Overwrite $\pi_h(\tau_h) = a$ and define $\tau_{h+1} = (\tau_h, a, c, s')$. Then, by definition of the interaction with M, $\mathbb{P}^{\pi}[\tau_{h+1}] \geq \mathbb{P}^{\pi}[\tau_h]\pi_h(a \mid \tau_h)C_h(c \mid s, a)P_h(s' \mid s, a) > 0$. Here we used the fact that if $\mathbb{P}^{\pi}_{\tau_h}[s_h = s, \bar{c}_h = \bar{c}] = 1$ then $\mathbb{P}^{\pi}[\tau_h] > 0$ by the definition of conditional probability. Thus, $\mathbb{P}^{\pi}_{\tau_{h+1}}[s_{h+1} = s', \bar{c}_{h+1} = \bar{c}'] = 1$. By assumption, $\mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \leq B]$ holds for all $k \in [h-1]$. For $k = h$, we have

$$\begin{aligned}
\mathbb{P}^{\pi}_{\tau_h}[\bar{c}_{h+1} \leq B] &= \mathbb{P}^{\pi}_{\tau_h}[\bar{c}_h + c_h \leq B \mid s_h = s, \bar{c}_h = \bar{c}] \\
&= \sum_{a'} \pi_h(a' \mid \tau_h) \Pr_{C_h(s,a')}[\bar{c} + c \leq B] \\
&= \Pr_{C_h(s,a)}[\bar{c} + c \leq B] \\
&= 1.
\end{aligned}$$

The first line used the law of total probability, the fact that $\mathbb{P}^{\pi}_{\tau_h}[s_h = s, \bar{c}_h = \bar{c}] = 1$, and the recursive decomposition of cumulative costs. The second line uses law of total probability on c_h. The third line follows since $\pi_h(a \mid \tau_h) = 1$ since $\pi_h(\tau_h) = a$ deterministically. The last line used the fact that $\Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1$. Thus, we see that $(s', \bar{c}') \in SE_{h+1}$. □

Claim 2. *For all $h \in [H+1]$, $\bar{\mathcal{S}}_h \supseteq SE_h$.*

Proof. We proceed by induction on h. For the base case, we consider $h = 1$. Observe that for any $\pi \in \Pi$, the only τ_1 that has non-zero probability is $\tau_1 = s_0$ since M starts at time 1 in state s_0. Also, $\bar{c}_1 = 0$ since no cost has been accrued by time 1. Thus, $SE_1 \subseteq \{(s_0, 0)\} = \bar{\mathcal{S}}_1$.

For the inductive step, we consider any $h \geq 1$. Let $(s', \bar{c}') \in SE_{h+1}$. By definition, there exists some $\pi \in \Pi$ and some $\tau \in \mathcal{H}$ satisfying,

$$\mathbb{P}^{\pi}_{\tau_{h+1}}[s_{h+1} = s', \bar{c}_{h+1} = \bar{c}'] = 1 \text{ and } \forall k \in [h],\ \mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \leq B] = 1.$$

Decompose $\tau_{h+1} = (\tau_h, a, c, s')$ where $s_h = s$ and $\bar{c}_h = \bar{c}$. Since $\mathbb{P}^{\pi}_{\tau_{h+1}}[s_{h+1} = s', \bar{c}_{h+1} = \bar{c}'] = 1$, we observe that

$$0 < \mathbb{P}^{\pi}[\tau_{h+1}] = \mathbb{P}^{\pi}[\tau_h]\pi_h(a \mid \tau_h)C_h(c \mid s, a)P_h(s' \mid s, a).$$

Thus, $\mathbb{P}^{\pi}_{\tau_h}[s_h = s, \bar{c}_h = \bar{c}] = 1$. Also, we immediately know that $\mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \leq B]$ $\forall k \in [h-1]$ since any sub-history of τ_h is also a sub-history of τ_{h+1}. Hence, $(s, \bar{c}) \in SE_h$ and so the induction hypothesis implies that $(s, \bar{c}) \in \bar{\mathcal{S}}_h$. We have already seen that $\bar{c}' = \bar{c} + c$, $C_h(c \mid s, a) > 0$ and $P_h(s' \mid s, a)$. To show that $(s', \bar{c}') \in \bar{\mathcal{S}}_{h+1}$, it then suffices to argue that $\Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1$. To this end, observe as in Claim 1 that,

$$1 = \mathbb{P}^{\pi}_{\tau_h}[\bar{c}_{h+1} \leq B] = \sum_{a'} \pi_h(a' \mid \tau_h) \Pr_{C_h(s,a')}[\bar{c} + c \leq B].$$

This implies that for all a', $\Pr_{C_h(s,a')}[\bar{c} + c \leq B] = 1$, otherwise we would have,

$$\sum_{a'} \pi_h(a' \mid \tau_h) \Pr_{C_h(s,a')}[\bar{c} + c \leq B] < \sum_{a'} \pi_h(a' \mid \tau_h) = 1,$$

which is a contradiction. Thus, $c + C_h(s,a) \leq B$ and so $(s', \bar{c}') \in \bar{\mathcal{S}}_{h+1}$. □

Proof of Lemma 7.

Proof.

First Claim. Fix any $(s, \bar{c})$ and suppose that $\mathbb{P}^{\pi}_{M}[s_h = s, \bar{c}_h = \bar{c}] > 0$ where $\pi \in \Pi_M$. Since $\pi \in \Pi_M$, we know that $\mathbb{P}^{\pi}_{M}\left[\forall k \in [H] \sum_{t=1}^{k} c_t \leq B\right] = 1$. Since the history distribution has finite support whenever the cost distributions do, we see for any history $\tau_{h+1} \in \mathcal{H}_{h+1}$ with $\mathbb{P}^{\pi}_{M}[\tau_{h+1}] > 0$ it must be the case that $\bar{c}_{h+1} \leq B$. In fact, it must also be the case that $\Pr_{c \sim C_h(s_h, a_h)}[c + \bar{c}_h \leq B] = 1$ otherwise there exists a realization of c and $\bar{c}_h$ under π for which the anytime constraint would be violated.

Moreover, this must hold for any subhistory of τ_{h+1} since those are also realized with non-zero probability under π. In symbols, we see that $\mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \leq B] = 1$ for all $k \in [h]$. Thus, $(s, \bar{c}) \in SE_h$. Since $(s, \bar{c})$ was arbitrary, we conclude by Lemma 20 that $\bar{\mathcal{S}}_h = SE_h \supseteq \mathcal{F}_h$.

observe that $|\bar{\mathcal{S}}_1| = 1$. By the inductive definition of $\bar{\mathcal{S}}_{h+1}$, we see that for any $(s, \bar{c}) \in \bar{\mathcal{S}}_h$, $(s, \bar{c})$ is responsible for adding at most $S \sum_{a \in \mathcal{A}} |C_h(s,a)| \leq SAn$ pairs $(s', \bar{c}')$ into $\bar{\mathcal{S}}_{h+1}$. Specifically, each next state s', current action a, and current cost $c \in C_h(s,a)$ yields at most one new element of $\bar{\mathcal{S}}_{h+1}$. Thus, $|\bar{\mathcal{S}}_{h+1}| \leq SAn|\bar{\mathcal{S}}_h|$. Since $S, A, n < \infty$, we see inductively that $|\bar{\mathcal{S}}_h| < \infty$ for all $h \in [H+1]$.

Second Claim. Suppose that for each (h, s) there is some a with $C_h(s,a) = \{0\}$. For any $(s, \bar{c}) \in SE_{h+1}$, we know that there exists some π and τ_{h+1} for which $\mathbb{P}^{\pi}_{\tau_k}[\bar{c}_{k+1} \leq B] = 1$ for all $k \in [h]$. Now define the deterministic policy π' by $\pi'_k(\tau_k) = \pi_k(\tau_k)$ for all subhistories τ_k of τ_{h+1}, and $\pi'_k(\tau_k) = a$ for any a with $C_k(s_k, a) = \{0\}$ otherwise. Clearly, π' never accumulates more cost than a subhistory

of τ_{h+1} since it always takes 0 cost actions after inducing a different history than one contained in τ_{h+1}.

Since under any subhistory of τ_{h+1}, π' satisfies the constraints by definition of SE_{h+1}, we know that $\pi' \in \Pi_M$. We also see that $\mathbb{P}_M^{\pi'}[s_{h+1} = s, \bar{c}_{h+1} = \bar{c}] > 0$ and so $(s, \bar{c}) \in \mathcal{F}_{h+1}$. Since $(s, \bar{c})$ was arbitrary we have that $SE_{h+1} = \mathcal{F}_{h+1}$. As h was arbitrary the claim holds. □

Observation 4. *For all $h > 1$, if $(s, \bar{c}) \in \bar{\mathcal{S}}_h$, then $\bar{c} \leq B$.*

Proof. For any $h \geq 1$, any $(s, \bar{c}) \in \bar{\mathcal{S}}_{h+1}$ satisfies $\bar{c}' = \bar{c} + c$ where $c \in C_h(s, a)$ and $\Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1$. Since $C_h(s, a)$ has finite support, this means that for any such $c \in C_h(s, a)$, we have that $\bar{c} + c \leq B$. In particular, $\bar{c}' = \bar{c} + c \leq B$. □

C.2.2 Proof of Lemma 8

The tabular *policy evaluation equations* (Equation 4.2.6 [91]) naturally translate to the cost setting as follows:

$$V_h^\pi(\tau_h) = \sum_{a \in \mathcal{A}} \pi_h(a \mid \tau_h) \left(r_h(s, a) + \sum_c \sum_{s'} C_h(c \mid s, a) P_h(s' \mid s, a) V_{h+1}^\pi(\tau_h, a, c, s') \right).$$

We can write this more generally in the compact form:

$$V_h^\pi(\tau_h) = \mathbb{E}_{\tau_h}^\pi \left[r_h(s, a) + \mathbb{E}_{\tau_{h+1}}^\pi \left[V_{h+1}^\pi(\tau_{h+1}) \right] \right]. \tag{PE}$$

The classic *Bellman Optimality Equations* (Equation 4.3.2 [91]) are,

$$\mathcal{V}_h^*(s) = \max_{a \in \mathcal{A}(s)} r_h(s, a) + \mathbb{E}_{s'} \left[\mathcal{V}_{h+1}^*(s') \right]$$

Observe that the optimality equations for $\bar{M}$ are,

$$\mathcal{V}_h^*(\bar{s}) = \max_{\bar{a} \in \bar{\mathcal{A}}_h(\bar{s})} \bar{r}_h(\bar{s}, \bar{a}) + \mathbb{E}_{\bar{s}'}\left[\mathcal{V}_{h+1}^*(\bar{s}')\right],$$

which reduce to

$$\mathcal{V}_h^*(s, \bar{c}) = \max_{a : \Pr_{c \sim C_h(s,a)}[\bar{c}+c \leq B]=1} r_h(s, a) + \mathbb{E}_{c,s'}\left[\mathcal{V}_{h+1}^*(s', \bar{c}+c)\right], \tag{BE}$$

where $\mathcal{V}_{H+1}^*(s, \bar{c}) = 0$. We then define $\mathcal{V}_h^*(s, \bar{c}) = \sup_\pi V_h^\pi(s, \bar{c})$. Note, if π chooses any action a for which $a \notin \bar{\mathcal{A}}(s, \bar{c})$, then $V_h^\pi(s, \bar{c}) := -\infty$ and we call π infeasible for $\bar{M}$.

Observation 5. *For any $\tau_h \in W_h(s, \bar{c})$, if $a \in \mathcal{A}$ satisfies $\Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1$, $s' \in \mathcal{S}$ satisfies $P_h(s' \mid s, a) > 0$, and $c \in C_h(s, a)$, then for $\tau_{h+1} := (\tau_h, a, c, s')$, $\tau_{h+1} \in W_{h+1}(s', \bar{c} + c)$.*

Proof. If $\tau_h \in W_h(s, \bar{c})$, then there exists some $\pi \in \Pi$ with,

$$\mathbb{P}_{\tau_h}^\pi\left[s_h = s, \bar{c}_h = \bar{c}\right] = 1 \text{ and } \forall k \in [h-1],\ \mathbb{P}_{\tau_k}^\pi\left[\bar{c}_{k+1} \leq B\right] = 1.$$

Define $\pi_h(\tau_h) = a$. Immediately, we see,

$$\mathbb{P}^\pi[\tau_{h+1}] = \mathbb{P}^\pi[\tau_h]\pi_h(a \mid \tau_h)C_h(c \mid s, a)P_h(s' \mid s, a) > 0,$$

so $\mathbb{P}_{\tau_{h+1}}^\pi[s_{h+1} = s', \bar{c}_{h+1} = \bar{c} + c] = 1$. Also, $\mathbb{P}_{\tau_h}^\pi[\bar{c}_{h+1} \leq B] = \Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1$ using the same argument as in Claim 1. Thus, $\tau_{h+1} \in W_{h+1}(s', \bar{c} + c)$. □

Now, we split the proof of Lemma 8 into two claims.

Claim 3. *For any $h \in [H+1]$, if $(s, \bar{c}) \in \bar{\mathcal{S}}_h$ and $\tau_h \in W_h(s, \bar{c})$, then $\mathcal{V}_h^*(s, \bar{c}) \geq V^*(\tau_h)$.*

Proof. We proceed by induction on h. For the base case, we consider $h = H + 1$. Let $(s, \bar{c}) \in \bar{\mathcal{S}}_{H+1}$ and $\tau_{H+1} \in W_{H+1}(s, \bar{c})$. By definition, $\mathcal{V}^*_{H+1}(s, \bar{c}) = 0$. If $\Pi_M(\tau_{H+1}) = \varnothing$, then $V^*(\tau_{H+1}) = -\infty < 0 = \mathcal{V}^*_{H+1}(s, \bar{c})$. Otherwise, for any $\pi \in \Pi_M(\tau_{H+1})$, $V^\pi_{H+1}(\tau_{H+1}) = 0$ by definition. Since π was arbitrary we see that $V^*(\tau_{H+1}) = 0 \leq 0 = \mathcal{V}^*_{H+1}(s, \bar{c})$.

For the inductive step, we consider $h \leq H$. Fix any $(s, \bar{c}) \in \bar{\mathcal{S}}_h$ and let $\tau_h \in W_h(s, \bar{c})$. If $\Pi_M(\tau_h) = \varnothing$, then $V^*(\tau_h) = -\infty \leq \mathcal{V}^*_h(s, \bar{c})$. Otherwise, fix any $\pi \in \Pi_M(\tau_h)$. Suppose $\tau_{h+1} = (\tau_h, a, c, s')$ where $\pi_h(a \mid \tau_h) > 0$, $C_h(c \mid s, a) > 0$, and $P_h(s' \mid s, a) > 0$. For any full history $\tau \in \mathcal{H}$ satisfying $\mathbb{P}^\pi_{\tau_{h+1}}[\tau] > 0$, we have $\mathbb{P}^\pi_{\tau_h}[\tau] = \mathbb{P}^\pi_{\tau_{h+1}}[\tau]\mathbb{P}^\pi_{\tau_h}[\tau_{h+1}] > 0$. Since $\pi \in \Pi_M(\tau_h)$, we know that for all complete histories $\tau \in \mathcal{H}$ with $\mathbb{P}^\pi_{\tau_h}[\tau] > 0$ that $\bar{c}_{k+1} \leq B$ for all $k \in [H]$. Consequently, for any $\tau \in \mathcal{H}$ satisfying $\mathbb{P}^\pi_{\tau_{h+1}}[\tau] > 0$, $\bar{c}_{k+1} \leq B$ for all $k \in [H]$. This means that $\mathbb{P}^\pi_{\tau_{h+1}}[\bar{c}_{k+1} \leq B] = 1$ for all $k \in [H]$ and so $\pi \in \Pi_M(\tau_{h+1})$.

By (BE),

$$
\begin{aligned}
\mathcal{V}^*_h(s, \bar{c}) &= \max_{a : \Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1} r_h(s, a) + \mathbb{E}_{c, s'}\left[\mathcal{V}^*_{h+1}(s', \bar{c} + c)\right] \\
&\geq \max_{a : \Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1} r_h(s, a) + \mathbb{E}_{c, s'}\left[V^*_{h+1}(\tau_{h+1})\right] \\
&\geq \sum_{a : \Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1} \pi_h(a \mid \tau_h)\left(r_h(s, a) + \mathbb{E}_{\tau_{h+1}}\left[V^*_{h+1}(\tau_{h+1})\right]\right) \\
&\geq \sum_{a : \Pr_{c \sim C_h(s,a)}[\bar{c} + c \leq B] = 1} \pi_h(a \mid \tau_h)\left(r_h(s, a) + \mathbb{E}_{\tau_{h+1}}\left[V^\pi_{h+1}(\tau_{h+1})\right]\right) \\
&= \sum_{a \in \mathcal{A}} \pi_h(a \mid \tau_h)\left(r_h(s, a) + \mathbb{E}_{\tau_{h+1}}\left[V^\pi_{h+1}(\tau_{h+1})\right]\right) \\
&= V^\pi_h(\tau_h).
\end{aligned}
$$

The second line follows from the induction hypothesis, where $\tau_{h+1} = (\tau_h, a, c, s') \in W_{h+1}(s', \bar{c} + c)$ by Observation 5. The third line follows since the finite maximum

is larger than the finite average (see Lemma 4.3.1 of [91]). The fourth line follows since $\pi \in \Pi_M(\tau_{h+1})$ implies $V^*(\tau_{h+1}) \geq V^\pi_{h+1}(\tau_h)$. $\Pr_{c\sim C_h(s,a)}[\bar{c} + c \leq B] = 1$. The fifth line follows since π cannot place non-zero weight on any action a satisfying $\bar{c} + C_h(s,a) > B$. Otherwise, we would have,

$$\mathbb{P}^\pi_{\tau_h}[\bar{c}_{h+1} > B] \geq \pi_h(a \mid \tau_h) \Pr_{c\sim C_h(s,a)}[\bar{c} + c > B] > 0,$$

contradicting that $\pi \in \Pi_M(\tau_h)$. The final line uses (PE).

Since π was arbitrary, we see that $\mathcal{V}^*_h(s,\bar{c}) \geq V^*(\tau_h)$.

□

Claim 4. *For any $h \in [H+1]$, if $(s,\bar{c}) \in \bar{\mathcal{S}}_h$ and $\tau_h \in W_h(s,\bar{c})$, then $\mathcal{V}^*_h(s,\bar{c}) \leq V^*(\tau_h)$.*

Proof. We proceed by induction on h. For the base case, we consider $h = H+1$. If $(s,\bar{c}) \in \bar{\mathcal{S}}_{H+1}$ and $\tau_{H+1} \in W_{H+1}(s,\bar{c})$, then by definition there exists some $\pi \in \Pi$ for which $\mathbb{P}^\pi[\tau_{H+1}] > 0$ and $\mathbb{P}^\pi_{\tau_k}[\bar{c}_{k+1} \leq B] = 1$ for all $k \in [H]$. We saw in the proof of Claim 2 that for any $k \leq H$, $(s_{k+1}, \bar{c}_{k+1}) \in \bar{\mathcal{S}}_{k+1}$. Thus, by Observation 4, we have $\bar{c}_{k+1} \leq B$ for all $k \in [H]$ which implies that $\mathbb{P}^\pi_{\tau_{H+1}}[\bar{c}_{k+1} \leq B] = 1$ for all $k \in [H]$. Hence, $\pi \in \Pi_M(\tau_{H+1})$ is a feasible solution to the optimization defining $V^*(\tau_{H+1})$ implying that $V^*(\tau_{H+1}) = 0$. By definition, we also have $\mathcal{V}^*_{H+1}(s,\bar{c}) = 0 \leq V^*(\tau_{H+1})$.

For the inductive step, we consider $h \leq H$. Let $(s,\bar{c}) \in \bar{\mathcal{S}}_h$ and $\tau_h \in W_h(s,\bar{c})$. Consider the deterministic optimal partial policy $\bar{\pi}$ for $\bar{M}$ defined by solutions to (BE). Formally, for all $t \geq h$,

$$\bar{\pi}_t(s,\bar{c}) \in \operatorname*{arg\,max}_{a:\Pr_{C_t(s,a)}[\bar{c}+c\leq B]=1} r_t(s,a) + \mathbb{E}_{c,s'}\left[\mathcal{V}^*_{t+1}(s', \bar{c}+c)\right].$$

If there is no feasible action for any of these equations of form $(t, s', \bar{c}')$ where

$t \geq h$ and $(s', \bar{c}') \in \bar{\mathcal{S}}_t$ are reachable from $(h, s, \bar{c})$ with non-zero probability, then $\mathcal{V}_h^*(s, \bar{c}) = -\infty$. In this case, clearly $\mathcal{V}_h^*(s, \bar{c}) \leq V^*(\tau_h)$. Otherwise, suppose solutions to (BE) exist so that $\bar{\pi}$ is well-defined from $(s, \bar{c})$ at time h onward. It is well known (see Theorem 4.3.3 from [91]) that $\mathcal{V}_t^{\bar{\pi}}(s', \bar{c}') = \mathcal{V}_t^*(s', \bar{c}')$ for all $t \geq h$ and all $(s', \bar{c}') \in \bar{\mathcal{S}}_t$. We unpack $\bar{\pi}$ into a partial policy π for M defined by,

$$\pi_t(\tau_t) = \begin{cases} \bar{\pi}_t(s_t, \bar{c}_t) & \text{if } (s_t, \bar{c}_t) \in \bar{\mathcal{S}}_t \\ a_1 & \text{o.w.} \end{cases}$$

Here, a_1 is an arbitrary element of $\mathcal{A}$. To make π a full policy, we can define π_t arbitrarily for any $t < h$.

We first show that for all $t \geq h$, $\mathbb{P}^{\pi}_{\tau_h}[(s_t, \bar{c}_t) \in \bar{\mathcal{S}}_t] = 1$. We proceed by induction on t. For the base case, we consider $t = h$. By assumption, $\tau_h \in W_h(s, \bar{c})$ so $(s_h, \bar{c}_h) = (s, \bar{c}) \in \bar{\mathcal{S}}_h$. Thus, $\mathbb{P}^{\pi}_{\tau_h}[(s_h, \bar{c}_h) \in \bar{\mathcal{S}}_h] = \mathbb{P}^{\pi}_{\tau_h}[(s, \bar{c}) \in \bar{\mathcal{S}}_h] = 1$.

For the inductive step, we consider $t \geq h$. By the induction hypothesis, we know that $\mathbb{P}^{\pi}_{\tau_h}[(s_t, \bar{c}_t) \in \bar{\mathcal{S}}_t] = 1$. By the law of total probability, it is then clear that,

$$\begin{aligned} &\mathbb{P}^{\pi}_{\tau_h}[(s_{t+1}, \bar{c}_{t+1}) \in \bar{\mathcal{S}}_{t+1}] = \mathbb{P}^{\pi}_{\tau_h}[(s_{t+1}, \bar{c}_{t+1}) \in \bar{\mathcal{S}}_{t+1} \mid (s_t, \bar{c}_t) \in \bar{\mathcal{S}}_t] \\ &= \sum_{(s', \bar{c}') \in \bar{\mathcal{S}}_t} \mathbb{P}^{\pi}_{\tau_h}[(s_{t+1}, \bar{c}_{t+1}) \in \bar{\mathcal{S}}_{t+1} \mid s_t = s', \bar{c}_t = \bar{c}'] \mathbb{P}^{\pi}_{\tau_h}[s_t = s', \bar{c}_t = \bar{c}']. \end{aligned}$$

Above we have used the fact that for any $(s', \bar{c}') \in \bar{\mathcal{S}}_t$, the event that $\{s_t = s', \bar{c}_t = \bar{c}', (s_t, \bar{c}_t) \in \bar{\mathcal{S}}_t\} = \{s_t = s', \bar{c}_t = \bar{c}'\}$.

For any τ_t with $(s_t, \bar{c}_t) = (s', \bar{c}') \in \bar{\mathcal{S}}_t$, by definition, $\pi_t(\tau_t) = \bar{\pi}_t(s', \bar{c}') = a' \in \{a \in \mathcal{A} \mid \bar{c}' + C_t(s', a) \leq B\}$. By the inductive definition of $\bar{\mathcal{S}}_{t+1}$, we then see that $(s'', \bar{c}' + c') \in \bar{\mathcal{S}}_{t+1}$ for any $s'' \sim P_t(s', a')$ and $c' \sim C_t(s', a')$. Hence, $\mathbb{P}^{\pi}_{\tau_h}[(s_{t+1}, \bar{c}_{t+1}) \in$

$\bar{\mathcal{S}}_{t+1} \mid s_t = s', \bar{c}_t = \bar{c}'] = 1$. We then see that,

$$\begin{aligned}\mathbb{P}^{\pi}_{\tau_h}[(s_{t+1}, \bar{c}_{t+1}) \in \bar{\mathcal{S}}_{t+1}] &= \sum_{(s',\bar{c}') \in \bar{\mathcal{S}}_t} \mathbb{P}^{\pi}_{\tau_h}[s_t = s', \bar{c}_t = \bar{c}'] \\ &= \mathbb{P}^{\pi}_{\tau_h}[(s_t, \bar{c}_t) \in \bar{\mathcal{S}}_t] \\ &= 1\end{aligned}$$

This completes the induction.

Since under τ_h, π induces only histories whose state and cumulative cost are in $\bar{\mathcal{S}}$, we see that π's behavior is identical to $\bar{\pi}$'s almost surely. In particular, it is easy to verify by induction using (PE) and Observation 5 that,

$$\begin{aligned}V^{\pi}_h(\tau_h) &= \mathbb{E}^{\pi}_{\tau_h}\left[r_h(s,a) + E_{\tau_{h+1}}\left[V^{\pi}_{h+1}(\tau_{h+1})\right]\right] \\ &= \mathbb{E}^{\bar{\pi}}_{(s,\bar{c})}\left[r_h((s,\bar{c}),a) + E_{(s',\bar{c}')}\left[\mathcal{V}^{\bar{\pi}}_{h+1}(s',\bar{c}')\right]\right] \\ &= V^{\bar{\pi}}_h(s,\bar{c}) \\ &= \mathcal{V}^{*}_h(s,\bar{c}).\end{aligned}$$

By Observation 4, we see if $(s_{k+1}, \bar{c}_{k+1}) \in \bar{\mathcal{S}}_{k+1}$ then $\bar{c}_{k+1} \leq B$. It is then clear by monotonicity of probability that $\mathbb{P}^{\pi}_{\tau_h}[c_{k+1} \leq B] \geq \mathbb{P}^{\pi}_{\tau_h}[(s_{k+1}, c_{k+1}) \leq \bar{\mathcal{S}}_{k+1}] = 1$ for all $k \in [H]$. Hence, $\pi \in \Pi_M(\tau_h)$ and so $\mathcal{V}^{*}_h(s,\bar{c}) = V^{\pi}_h(\tau_h) \leq V^{*}(\tau_h)$.

□

Observation 6 (Cost-Augmented Probability Measures)**.** *We note we can treat $\bar{\pi}$ defined in the proof of Claim 4 as a history dependent policy in the same way we defined π. Doing this induces a probability measure over histories. We observe that measure is identical as the one induced by the true history-dependent policy π. Thus, we can directly use augmented policies with M and reason about their values and*

costs with respect to M.

C.2.3 Proof of Theorem 8

Proof. From Lemma 8, we see that $V^* = V^*(s_0) = \mathcal{V}_1^*(s_0, 0) = \mathcal{V}^*$. Furthermore, in Claim 4, we saw the policy defined by the optimality equations (BE) achieves the optimal value, $\mathcal{V}^{\bar{\pi}} = \mathcal{V}^* = V^*$. Furthermore, $\bar{\pi}$ behaves identically to a feasible history-dependent policy π almost surely. In particular, as argued in Claim 4 both policies only induce cumulative costs appearing in $\bar{\mathcal{S}}_h$ at any time h and so by Observation 4 we know that both policies' cumulative costs are at most B anytime. □

C.2.4 Proof of Corollary 2

The theorem follows immediately from Theorem 8 and the argument from the main text.

C.2.5 Proof of Proposition 10

Proof. By definition of $\bar{\mathcal{S}}$, it is clear that $|\bar{\mathcal{S}}_h| \leq |\mathcal{S}|\mathcal{D}$, and by inspection, we see that $|\bar{\mathcal{A}}| \leq |\mathcal{A}|$. The agent can construct $\bar{\mathcal{S}}$ using our forward induction procedure, Algorithm 18, in $O(\sum_{h=1}^{H-1} |\bar{\mathcal{S}}_h|SAn) = O(HS^2An\mathcal{D})$ time. Also, the agent can compute $\bar{P}$ by forward induction in the same amount of time so long as the agent only records the non-zero transitions. Thus, $\bar{M}$ can be computed in $O(HS^2An\mathcal{D})$ time.

1. By directly using backward induction on $\bar{M}$ [91], we see that an optimal policy can be computed in $O(H|\bar{\mathcal{S}}|^2|\bar{\mathcal{A}}|) = O(HS^2A\mathcal{D}^2)$ time. However, this analysis can be refined: for any sub-problem of the backward induction $(h, (s, \bar{c}))$ and any action a, there are at most nS state-cost pairs that can be reached in

the next period (namely, those of the form $(s', \bar{c} + c)$) rather than $S\mathcal{D}$. Thus, backward induction runs in $O(HS^2An\mathcal{D})$ time, and so planning in total can be performed in $O(HS^2An\mathcal{D})$ time.

2. Similarly, PAC (probably-approximately correct) learning can be done with sample complexity $\tilde{O}(H^3|\bar{\mathcal{S}}||\bar{\mathcal{A}}|\log(\frac{1}{\delta})/\gamma^2) = \tilde{O}(H^3SA\mathcal{D}\log(\frac{1}{\delta})/\gamma^2)$ [81], where δ is the confidence and γ is the accuracy. Note, we are translating the guarantee to the non-stationary state set setting which is why the $|\bar{\mathcal{S}}|$ term becomes $S\mathcal{D}$ instead of $HS\mathcal{D}$.

□

C.2.6 Proof of Lemma 9

Proof. Suppose each cost is represented with k bits of precision. For simplicity, we assume that k includes a possible sign bit. By ignoring insignificant digits, we can write each number in the form $2^{-i}b_{-i} + \ldots 2^{-1}b_{-1} + 2^0 b_0 + \ldots + 2^{k-i-1}b_{k-i}$ for some i. By dividing by 2^{-i}, each number is of the form $2^0 b_0 + \ldots + 2^{k-1}b_{k-1}$. Notice, the largest possible number that can be represented in this form is $\sum_{i=0}^{k-1} 2^i = 2^k - 1$. Since at each time h, we potentially add the maximum cost, the largest cumulative cost ever achieved is at most $2^k H - 1$. Since that is the largest cost achievable, no more than $2^k H$ can ever be achieved through all H times. Similarly, no cost can be achieved smaller than $-2^k H$.

Thus each cumulative cost is in the range $[-2^k H + 1, 2^k H - 1]$ and so at most $2^{k+1}H$ cumulative costs can ever be created. By multiplying back the 2^{-i} term, we see at most $2^{k+1}H$ costs are ever generated by numbers with k bits of precision. Since this argument holds for each constraint independently, the total number of cumulative cost vectors that could ever be achieved is $(H2^{k+1})^d$. Hence, $\mathcal{D} \leq H^d 2^{(k+1)d}$. □

C.2.7 Proof of Theorem 9

Theorem 9 follows immediately from Proposition 10, Lemma 9, and the definition of fixed-parameter tractability [33].

C.3 Proofs for Section 4.4

For any h we let $\hat{c}_{h+1} := f(\tau_{h+1})$ be a random variable of the history defined inductively by $\hat{c}_1 = 0$ and $\hat{c}_{k+1} = f_k(\hat{c}_k, c_k)$ for all $k \leq h$. Notice that since f is a deterministic function, $\hat{c}_k$ can be computed from τ_{h+1} for all $k \in [h+1]$. Then, a probability distribution over $\hat{c}$ is induced by the one over histories. As such, approximate-cost augmented policies can also be viewed as history-dependent policies for M as in Observation 6.

C.3.1 Proof of Lemma 10

Proof. We proceed by induction on h. Fix any feasible policy π for $\hat{M}$. For the base case, we consider $h = 1$. By definition, $\bar{c}_1 = 0 = \hat{c}_1$ and so the claim trivially holds. For the inductive step, we consider any $h \geq 1$. By the induction hypothesis, we know that $\hat{c}_h \leq \bar{c}_h \leq \hat{c}_h + (h-1)\ell$ or $\hat{c}_h, \bar{c}_h \leq B - (H-h+1)c_{max}$ almost surely. We split the proof into cases.

1. First, suppose that $\hat{c}_h \leq \bar{c}_h \leq \hat{c}_h + (h-1)\ell$.

   (a) Furthermore, suppose that $\hat{c}_h + c_h \geq B - (H-h)c_{max}$ so that $\hat{c}_{h+1} = f_h(\hat{c}_h, c_h) = \hat{c}_h + \left\lfloor \frac{c_h}{\ell} \right\rfloor \ell$. By definition of the floor function, $\left\lfloor \frac{c_h}{\ell} \right\rfloor \leq \frac{c_h}{\ell}$. Thus,

$$\hat{c}_{h+1} \leq \hat{c}_h + \frac{c_h}{\ell}\ell = \hat{c}_h + c_h \leq \bar{c}_h + c_h = \bar{c}_{h+1},$$

holds almost surely, where we used the inductive hypothesis with our case assumption to infer that $\hat{c}_h \leq \bar{c}_h$ almost surely in the second inequality. Also, by definition of the floor function, $\frac{c_h}{\ell} \leq \left\lfloor \frac{c_h}{\ell} \right\rfloor + 1$. We then see that,

$$\bar{c}_{h+1} = \bar{c}_h + \frac{c_h}{\ell}\ell \leq \bar{c}_h + (\left\lfloor \frac{c_h}{\ell} \right\rfloor + 1)\ell \leq \hat{c}_h + (h-1)\ell + \left\lfloor \frac{c_h}{\ell} \right\rfloor \ell + \ell$$
$$= \hat{c}_{h+1} + h\ell.$$

The first inequality used the induction hypothesis with our case assumption and the second used the property of floors.

(b) Now, suppose that $\hat{c}_h + c_h < B - (H-h)c_{max}$ so that $\hat{c}_{h+1} = f_h(\hat{c}_h, c_h) = \left\lfloor \frac{B-(H-h)c_{max}}{\ell} \right\rfloor \ell$.

i. If $\bar{c}_{h+1} \leq \hat{c}_{h+1}$, then by definition we have,

$$\bar{c}_{h+1}, \hat{c}_{h+1} \leq \left\lfloor \frac{B-(H-h)c_{max}}{\ell} \right\rfloor \ell \leq B - (H-h)c_{max},$$

and we are done.

ii. Otherwise, if $\hat{c}_{h+1} \leq \bar{c}_{h+1}$, then we see that,

$$\bar{c}_{h+1} = \bar{c}_h + c_h \leq \hat{c}_h + (h-1)\ell + c_h < B - (H-h)c_{max} + (h-1)\ell$$
$$\leq (\left\lfloor \frac{B-(H-h)c_{max}}{\ell} \right\rfloor + 1)\ell + (h-1)\ell = \hat{c}_{h+1} + h\ell,$$

where the first inequality used the induction hypothesis with our case assumption.

2. Lastly, suppose that $\bar{c}_h, \hat{c}_h \leq B - (H-h+1)c_{max}$. Then, it is clear that,

$$\bar{c}_{h+1} = \bar{c}_h + c_h \leq \bar{c}_h + c_{max} \leq B - (H-h+1)c_{max} + c_{max} = B - (H-h)c_{max}.$$

Similarly, we see that either,

$$\hat{c}_{h+1} = \hat{c}_h + \left\lfloor \frac{c_h}{\ell} \right\rfloor \ell \leq \hat{c}_h + c_h \leq \hat{c}_h + c_{max} \leq B - (H-h+1)c_{max} + c_{max}$$
$$= B - (H-h)c_{max},$$

or,

$$\hat{c}_{h+1} = \left\lfloor \frac{B-(H-h)c_{max}}{\ell} \right\rfloor \ell \leq B - (H-h)c_{max}.$$

This completes the induction.

We next show the second claim. By definition, any approximate cost is an integer multiple of ℓ where the integer is in the range $\{\lfloor \frac{B-Hc_{max}}{\ell} \rfloor, \ldots, \lfloor \frac{B}{\ell} \rfloor\}$. The number of elements in this set is exactly,

$$\left\lfloor \frac{B}{\ell} \right\rfloor - \left\lfloor \frac{B-Hc_{max}}{\ell} \right\rfloor + 1 \leq \frac{B}{\ell} - (\frac{B-Hc_{max}}{\ell} - 1) + 1 = \frac{Hc_{max}}{\ell} + 2.$$

When there are d constraints, this analysis applies to each separately since we do vector operations component-wise. Thus, the total number of approximate costs is $(\frac{H\|c_{max}\|_\infty}{\ell} + 2)^d$.

□

C.3.2 Proof of Theorem 10

Proof. We first note that the same argument used to prove Theorem 8 immediately extends to the approximate MDP and implies that any feasible π for $\hat{M}$ satisfies $\mathbb{P}^\pi_{\hat{M}}[\forall t \in [H],\ \hat{c}_{t+1} \leq B] = 1$. Also, we note since $\hat{c}$ is a deterministic function of the history, we can view any policy π for $\hat{M}$ as a cost-history-dependent policy for M similar to in the proof of Observation 6. Thus, Lemma 10 implies that for any feasible π for $\hat{M}$ and any $h \in [H+1]$, $\mathbb{P}^\pi_M[\hat{c}_h \leq \bar{c}_h \leq \hat{c}_h + (h-1)\ell \vee \bar{c}_h, \hat{c}_h \leq B - (H-$

$h+1)c_{max}] = 1$. Since $\hat{c}_{h+1} \leq B$ a.s., we immediately see that $\mathbb{P}^{\pi}_{M}[\bar{c}_{h+1} \leq B + h\ell] = 1$ for all $h \in [H]$.

Furthermore, we observe that any feasible policy π for the anytime constraint is also feasible for $\hat{M}$ since $\mathbb{P}^{\pi}_{M}[\hat{c}_h \leq \bar{c}_h \vee \bar{c}_h, \hat{c}_h \leq B - (H-h+1)c_{max}] = 1$ implies that $\mathbb{P}^{\pi}_{M}[\hat{c}_{h+1} \leq B] = 1$ since $\bar{c}_{h+1} \leq B$ almost surely. Since the rewards of $\hat{M}$ only depends on the state and action, we see π achieves the same value in both MDPs. Thus, $\hat{V}^* \geq V^*$.

Lastly, Lemma 10 implies that $\mathcal{D}_{\hat{M}} \leq (\frac{H\|c_{max}\|_\infty}{\ell} + 2)^d$ which with Proposition 10 gives the storage complexity. □

C.3.3 Proof of Corollary 3

Proof. The proof is immediate from Theorem 10 and Proposition 10. □

C.3.4 Proof of Corollary 4

Proof. The proof is immediate from Theorem 10 and Proposition 10. □

C.3.5 Proof of Corollary 5

First observe that if $B < 0$ then the instance is trivially infeasible which can be determined in linear time. Otherwise, the immediate cost (in addition to the cumulative cost) induced by any feasible π is always in the range $[0, B]$. Specifically, the larger costs xB can never be accrued since there are no negative costs now to offset them, so we can effectively assume that $c_{max} \leq B$. Since the floor of any non-negative number is non-negative, the integer multiples of ℓ needed are now in the range $[0, \lfloor c_{max}/\ell \rfloor] \subseteq [0, \lfloor B/\ell \rfloor]$. Thus, we have $O(\frac{Hc_{max}}{\epsilon})$ approximate costs for the additive approximation since $\ell = \epsilon/H$, and $O(\frac{H}{\epsilon})$ approximate costs for the relative

approximation since $\ell = \epsilon B/H$. The complexities are reduced accordingly.

C.3.6 Proof of Proposition 11

Proof. Note that computing an optimal-value, ϵ-additive solution for the knapsack problem is equivalent to just solving the knapsack problem when $\epsilon < 1$. In particular, since each weight is integer-valued, if the sum of the weights is at most $B + \epsilon < B + 1$ then it is also at most B. By scaling the weights and budget by $\lceil 2\epsilon \rceil$, the same argument implies hardness for $\epsilon \geq 1$.

For relative approximations, we present a reduction from Partition to the problem of finding an optimal-value, ϵ-relative feasible solution to the knapsack problem with negative weights. Again, we focus on the $\epsilon < 1$ regime but note the proof extends using scaling. Let $X = \{x_1, \ldots, x_n\}$ be the set of positive integers input to the partition problem and $Sum(X) := \sum_{i=1}^{n} x_i$. Observe that $Sum(X)/2$ must be an integer else the instance is trivially a "No" instance. Define $v_i = 2x_i$ and $w_i = 2x_i$ for each $i \in [n]$. Also, we define a special item 0 with $v_0 = -Sum(X)$ and $w_0 = -Sum(X)$. We define the budget to be $B = 1$. We claim that there exists some $Y \subseteq [n]$ with $Sum(Y) = Sum(\overline{Y}) = Sum(X)/2$ if and only if there exists an $I \subseteq [n] \cup \{0\}$ with $\sum_{i \in I} v_i \geq 0$ and $\sum_{i \in I} w_i \leq B(1 + \epsilon)$.

- $[\implies]$ if Y is a solution to Partition, then we define $I = Y \cup 0$. We observe that,

$$\sum_{i \in I} v_i = -Sum(X) + 2 \sum_{i \in S} x_i = -Sum(X) + 2Sum(X)/2 = 0.$$

  Similarly, $\sum_{i \in I} w_i = 0 < 1 \leq B(1 + \epsilon)$. Thus, I satisfies the conditions.

- $[\impliedby]$ if I is an ϵ-relative feasible solution to Knapsack, observe that I must contain 0. In particular, each $w_i = 2x_i \geq 2 > (1 + \epsilon) = B(1 + \epsilon)$ and so for

approximate feasibility to hold it must be the case that a negative weight was included. Let $Y = I \setminus 0$. Then, we see that,

$$0 \leq \sum_{i \in I} v_i = -Sum(X) + 2 \sum_{i \in Y} x_i = -Sum(X) + 2Sum(Y).$$

Thus, $Sum(Y) \geq Sum(X)/2$. Similarly,

$$1 + \epsilon \geq \sum_{i \in I} w_i = -Sum(X) + 2Sum(Y).$$

Thus, $Sum(Y) \leq Sum(X)/2 + (1+\epsilon)/2 < Sum(X)/2 + 1$ since $\epsilon < 1$. Because $Sum(Y)$ is a sum of positive integers, and $Sum(X)/2$ is a positive integer, it must be the case that $Sum(Y) \leq Sum(X)/2$. Pairing this with $Sum(Y) \geq Sum(X)/2$ implies equality holds. Thus, Y is a solution to Partition.

Since the transformation can be made in linear time, we see that the reduction is polynomial time. Since Partition is NP-hard, we then see finding an optimal-value, ϵ-relative feasible solution to the knapsack problem with negative weights is NP-hard. □

C.3.7 Proof of Proposition 12

Proof. The proof is immediate from Corollary 3 and Corollary 4. □

C.4 Extensions

C.4.1 Generalized Anytime Constraints

Consider the constraints of the form,

$$\mathbb{P}_M^\pi \left[\forall k \in [H],\ \sum_{t=1}^{k} c_t \in [L_k, U_k] \right] = 1. \tag{C.2}$$

All of our exact methods carry over to this more general setting by simply tweaking the safe exploration set. In particular, we define,

$$\bar{\mathcal{S}}_{h+1} := \Big\{ (s', \bar{c}') \in \mathcal{S} \times \mathbb{R}^d \mid \exists (s, \bar{c}) \in \bar{\mathcal{S}}_h, \exists a \in \mathcal{A}, \exists c \in C_h(s,a), \\ \bar{c}' = \bar{c} + c,\ \Pr_{c \sim C_h(s,a)}[c + \bar{c} \in [L_h, U_h]] = 1,\ P_h(s' \mid s, a) > 0 \Big\}. \tag{C.3}$$

Similarly, each quantity in the analysis changes to consider the different intervals per time step. The proof is otherwise identical.

For the approximate methods, the additive results imply the costs are at most $U_k + \epsilon$ anytime, and since the costs are independent of the new restrictions, the complexity guarantees are the same. We could similarly give an approximation concerning the lower bound by using pessimistic costs. For the relative approximation, we now define ℓ with respect to $|U^{min}| = \min_k |U_k|$ and all costs should lie below $|U^{min}|$. The guarantees then translate over with $|U^{min}|$ taking the role of $|B|$.

C.4.2 General Almost-Sure Constraints

General almost-sure constraints require that,

$$\mathbb{P}_M^\pi \left[\sum_{t=1}^{H} c_t \leq B \right] = 1. \tag{C.4}$$

This can be easily captured by the generalized anytime constraints by making L_k smaller than kc_{min} and U_k larger than kc_{max} for any $k < H$ so that the process is effectively unconstrained until the last time step where $U_H = B$.

Observe then when applying our relative approximation, $U^{min} = U_H = B$ and so

the guarantees translate similarly as to the original anytime constraints. In particular, although $c_{max} \leq |B|$, the cumulative cost could be up to $H|B|$. This means the multiples of ℓ that need to be considered are in the set $\{\lfloor -xH^2/\epsilon \rfloor, \ldots, \lfloor xH^2/\epsilon \rfloor\}^d$. This changes the exact constants considered, but the asymptotic guarantees are the same. We do note however that the improvements in Corollary 5 do not extend to the general almost-sure case.

On the other hand, the additive approximation results now have $\|2Hc_{max} - B\|_\infty$ terms instead of $\|c_{max}\|_\infty$ terms. The asymptotic bounds then have $\|c_{max} - B/H\|_\infty$ terms.

C.4.3 Infinite Discounting

If the rewards and costs are discounted, it is easy to see that Theorem 8 still holds but the resultant MDP has infinite states and discontinuous reward function. However, our approximate methods work well. By simply using the horizon H to be the earliest time in which $\sum_{t=H+1}^{\infty} \gamma^t c_t \leq \epsilon$ almost surely, we can use our reduction to get an ϵ-additive feasible policy. Pairing this with our approximation algorithms gives a computationally efficient solution. To get a desired accuracy the effective horizon H may need to be increased before using the approximation algorithms.

Appendix D

Chapter 5 Appendix

D.1 Proofs for Section 5.2

D.1.1 Proof of Proposition 13

The proof follows from the standard proof of backward induction [91]. The main ideas for the proof can also be seen in the proof of Lemma 22 and Lemma 23.

D.1.2 Proof of Proposition 14

Proof.

1. (Expectation Constraints) We claim C_M^π captures expectation constraints. This is immediate as an expectation constraint takes the form $\mathbb{E}_M^\pi\left[\sum_h c_h(s_h, a_h)\right] \leq B$ and by definition $C_M^\pi = \mathbb{E}_M^\pi\left[\sum_{h=1}^H c_h(s_h, a_h)\right]$. Moreover, the standard policy evaluation equations for deterministic policies immediately imply,

$$C_h^\pi(\tau_h) = c_h(s, a) + \sum_{s'} P_h(s' \mid s, a) C_{h+1}^\pi(\tau_h, a, s'). \tag{EC}$$

Thus, (TR) holds. It is also easy to see that $\sum_{s'} P_h(s' \mid s, a) C_{h+1}^\pi(\tau_h, a, s')$ can

be computed recursively state-wise by,

$$P_h(1 \mid s, a)C^\pi_{h+1}(\tau_h, a, 1) + \sum_{s'=2}^{S} P_h(s' \mid s, a)C^\pi_{h+1}(\tau_h, a, s'), \tag{D.1}$$

and so (SR) holds. The infinity conditions and non-decreasing requirements are also easy to verify.

2. (Almost Sure Constraints) We claim that C^π_M captures almost sure constraints. This is because that for tabular MDPs, $\mathbb{P}^\pi_M[\sum_{h=1}^H c_h(s_h, a_h) \leq B] = 1$ if and only if for all $\tau \in \mathcal{H}_{H+1}$ with $\mathbb{P}^\pi_M[\tau] > 0$ it holds that $\sum_{h=1}^H c_h(s_h, a_h) \leq B$ if and only if $C^\pi_M = \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}^\pi_M[\tau]>0}} \sum_{h=1}^H c_h(s_h, a_h) \leq B$.

   Let $c(\tau) = \sum_{h=1}^H c_h(s_h, a_h)$ denote the cost of a full history $\tau \in \mathcal{H}_{H+1}$ and let $c_{h:t}(\tau) = \sum_{k=h}^t c_k(s_k, a_k)$ denote the partial cost of τ from time h to time t. Our choice of α and β imply that,

$$C^\pi_h(\tau_h) = c_h(s, a) + \max_{s' \in P_h(s,a)} C^\pi_{h+1}(\tau_h, a, s'). \tag{ASC}$$

   To show that C^π_M satisfies (TR), we prove for all $h \in [H+1]$ and all $\tau_h \in \mathcal{H}_h$ that

$$C_h(\tau_h) = \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}^\pi_M[\tau|\tau_h]>0}} c_{h:H}(\tau). \tag{D.2}$$

   Then, we see that $C^\pi_1(s_0) = \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}^\pi_M[\tau|s_0]>0}} c_{1:H}(\tau) = \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}^\pi_M[\tau]>0}} \sum_{h=1}^H c_h(s_h, a_h)$ $= C^\pi_M$. Thus, C^π_M satisfies (TR). Furthermore, it is clear that $\max_{s' \in P_h(s,a)}$ $C^\pi_{h+1}(\tau_h, a, s')$ can be computed state-recursively by,

$$\max(C^\pi_{h+1}(\tau_h, a, 1)[P_h(1 \mid s, a) > 0], \max_{s'=2}^{S} C^\pi_{h+1}(\tau_h, a, s')[P_h(s' \mid s, a) > 0]), \tag{D.3}$$

and so C_M^π satisfies (SR). The infinity conditions and non-decreasing requirements are also easy to verify.

We proceed by induction on h.

- (Base Case) For the base case, we consider $h = H + 1$. Observe that for any history τ, we have $c_{H+1:H}(\tau) = 0$ since it is an empty sum. Then, by definition of C_M^π, we see that $C_{H+1}^\pi(\tau_{H+1}) = 0 = \max_\tau 0 = \max_\tau c_{H+1:H}(\tau)$.
- (Inductive Step) For the inductive step, we consider $h \leq H$. Let $s = s_h(\tau_h)$ and $a = \pi_h(\tau_h)$. For any $\tau \in \mathcal{H}_{H+1}$ for which $\mathbb{P}_M^\pi[\tau \mid \tau_h] > 0$, we can decompose its cost by $c_{h:H}(\tau) = c_h(s, a) + c_{h+1:H}(\tau)$. Since a is fixed, we can remove $c_h(s, a)$ from the optimization to get,

$$\max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h:H}(\tau) = c_h(s, a) + \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h+1:H}(\tau).$$

  Next, we observe by the Markov property that $\mathbb{P}_M^\pi[\tau \mid \tau_h] = \sum_{s'} \mathbb{P}_M^\pi[\tau \mid \tau_h, a, s'] P_h(s' \mid s, a)$. Thus, $\mathbb{P}_M^\pi[\tau \mid \tau_h] > 0$ if and only if there exists some $s' \in P_h(s, a)$ satisfying $\mathbb{P}_M^\pi[\tau \mid \tau_h, a, s'] > 0$. This implies that,

$$\max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h+1:H}(\tau) = \max_{s' \in P_h(s,a)} \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h,a,s']>0}} c_{h+1:H}(\tau).$$

By applying the induction hypothesis, we see that,

$$\begin{aligned}\max_{\substack{\tau\in\mathcal{H}_{H+1}:\\ \mathbb{P}^\pi_M[\tau|\tau_h]>0}} c_{h:H}(\tau) &= c_h(s,a) + \max_{s'\in P_h(s,a)} \max_{\substack{\tau\in\mathcal{H}_{H+1}:\\ \mathbb{P}^\pi_M[\tau|\tau_h,a,s']>0}} c_{h+1:H}(\tau) \\ &= c_h(s,a) + \max_{s'\in P_h(s,a)} C^\pi_{h+1}(\tau_h,a,s') \\ &= C_h(\tau_h).\end{aligned}$$

The second line used the induction hypothesis and the third line used the definition of C^π_M.

3. (Anytime Constraints) We claim that C^π_M captures anytime constraints. This is because that for tabular MDPs, $\mathbb{P}^\pi_M[\forall t \in [H],\ \sum_{h=1}^t c_h(s_h,a_h) \le B] = 1$ if and only if for all $t \in [H]$ and $\tau \in \mathcal{H}_{H+1}$ with $\mathbb{P}^\pi_M[\tau] > 0$ it holds that $\sum_{h=1}^t c_h(s_h,a_h) \le B$ if and only if $C^\pi_M = \max_{t\in[H]} \max_{\tau\in\mathcal{H}_{H+1}:\mathbb{P}^\pi_M[\tau]>0} \sum_{h=1}^t c_h(s_h,a_h) \le B$.

Our choice of α and β imply that,

$$C^\pi_h(\tau_h) = c_h(s,a) + \max\left(0, \max_{s'\in P_h(s,a)} C^\pi_{h+1}(\tau_h,a,s')\right). \tag{AC}$$

To show that C^π_M satisfies (TR), we show that for all $h \in [H+1]$ and all $\tau_h \in \mathcal{H}_h$ that

$$C_h(\tau_h) = \max_{t\ge h} \max_{\substack{\tau\in\mathcal{H}_{H+1}:\\ \mathbb{P}^\pi_M[\tau|\tau_h]>0}} c_{h:t}(\tau). \tag{D.4}$$

Then, we see that $C^\pi_1(s_0) = \max_t \max_{\substack{\tau\in\mathcal{H}_{H+1}:\\ \mathbb{P}^\pi_M[\tau|s_0]>0}} c_{1:t}(\tau) = \max_{t\in[H]} \max_{\substack{\tau\in\mathcal{H}_{H+1}:\\ \mathbb{P}^\pi_M[\tau]>0}} \sum_{h=1}^t c_h(s_h,a_h) = C^\pi_M$. Thus, C^π_M satisfies (TR). Furthermore, it is clear that

$\max(0, \max_{s' \in P_h(s,a)} C^\pi_{h+1}(\tau_h, a, s'))$ can be computed state-recursively by,

$$\max\Big(\max(0, C^\pi_{h+1}(\tau_h, a, 1)[P_h(1 \mid s, a) > 0]), \max(0, \max_{s'=2}^{S} C^\pi_{h+1}(\tau_h, a, s')[P_h(s' \mid s, a) > 0])\Big), \tag{D.5}$$

and so C^π_M satisfies (SR). The infinity conditions and non-decreasing requirements are also easy to verify.

We proceed by induction on h.

- (Base Case) For the base case, we consider $h = H + 1$. Observe that for any history τ and t, we have $c_{H+1:t}(\tau) = 0$ since it is an empty sum. Then, by definition of C^π_M, we see that $C^\pi_{H+1}(\tau_{H+1}) = 0 = \max_t \max_\tau 0 = \max_t \max_\tau c_{H+1:t}(\tau)$[1].
- (Inductive Step) For the inductive step, we consider $h \leq H$. Let $s = s_h(\tau_h)$ and $a = \pi_h(\tau_h)$. By separately considering the case where $t = h$ and

[1]Technically, there is no $t \in [H]$ satisfying $t \geq H + 1$. We instead interpret the $t \geq h$ condition in the max as over all integers and define the immediate costs to be 0 for all future times to simplify the base case.

$t \geq h+1$ in the $\max_{t \geq h}$, we see that, $\max_{t \geq h} \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h:t}(\tau) =$

$$
\begin{aligned}
&\max \left(\max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h:h}(\tau), \max_{t \geq h+1} \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h:t}(\tau) \right) \\
&= \max \left(c_h(s,a), c_h(s,a) + \max_{t \geq h+1} \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h+1:t}(\tau) \right) \\
&= c_h(s,a) + \max \left(0, \max_{t \geq h+1} \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h]>0}} c_{h+1:t}(\tau) \right) \\
&= c_h(s,a) + \max \left(0, \max_{t \geq h+1} \max_{s' \in P_h(s,a)} \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h,a,s']>0}} c_{h+1:t}(\tau) \right) \\
&= c_h(s,a) + \max \left(0, \max_{s' \in P_h(s,a)} \max_{t \geq h+1} \max_{\substack{\tau \in \mathcal{H}_{H+1}: \\ \mathbb{P}_M^\pi[\tau|\tau_h,a,s']>0}} c_{h+1:t}(\tau) \right) \\
&= c_h(s,a) + \max \left(0, \max_{s' \in P_h(s,a)} C_{h+1}^\pi(\tau_h, a, s') \right) \\
&= C_h(\tau_h).
\end{aligned}
$$

The second line used the fact that $c_{h:h}(\tau) = c_h(s,a)$ and the recursive definition of $c_{h:t}(\tau)$. The fourth line used the result proven for the almost sure case above. The sixth line used the induction hypothesis. The last line used the definition of C_M^π.

□

D.2 Proofs for Section 5.3

D.2.1 Helpful Technical Lemmas

Here, we use a different, inductive definition for $\mathcal{V}$ then in the main text. However, the following lemma shows they are equivalent.

Definition 27 (Value Space)**.** For any $s \in \mathcal{S}$, we define $\mathcal{V}_{H+1}(s) \stackrel{\text{def}}{=} \{0\}$, and for any $h \in [H]$,

$$\mathcal{V}_h(s) \stackrel{\text{def}}{=} \bigcup_a \bigcup_{\mathbf{v} \in \times_{s'} \mathcal{V}_{h+1}(s')} \left\{ r_h(s,a) + \sum_{s'} P_h(s' \mid s,a) v_{s'} \right\}. \tag{D.6}$$

We define $\mathcal{V} \stackrel{\text{def}}{=} \bigcup_{h,s} \mathcal{V}_h(s)$.

Lemma 21 (Value Intution)**.** *For all $s \in \mathcal{S}$ and $h \in [H+1]$,*

$$\mathcal{V}_h(s) = \left\{ v \in \mathbb{R} \mid \exists \pi \in \Pi^D, \tau_h \in \mathcal{H}_h,\ (s = s_h(\tau_h) \wedge V_h^\pi(\tau_h) = v) \right\}, \tag{D.7}$$

and $|\mathcal{V}_h(s)| \leq A^{\sum_{t=h}^H S^{H-t}}$. Thus, $\mathcal{V}$ can be computed in finite time using backward induction.

Lemma 22 (Cost)**.** *For any $h \in [H+1]$, $\tau_h \in \mathcal{H}_h$, and $v \in \mathcal{V}$, if $s = s_h(\tau_h)$, then,*

$$\begin{aligned} \bar{C}_h^*(s,v) \leq \min_{\pi \in \Pi^D} & \ C_h^\pi(\tau_h) \\ & s.t.\ V_h^\pi(\tau_h) \geq v. \end{aligned} \tag{D.8}$$

Lemma 23 (Value)**.** *Suppose that $\pi \in \Pi^D$. For all $h \in [H+1]$ and $(s,v) \in \bar{\mathcal{S}}$, if $\bar{C}_h^\pi(s,v) < \infty$, then $\bar{V}_h^\pi(s,v) \geq v$.*

Remark 22 (Technical Subtlety). Technically, $V_h^\pi(\tau_h)$ is only well defined if $\mathbb{P}_M^\pi[\tau_h] > 0$ and all of our arguments technically should assume this is the case. However, it is standard in MDP theory to define the policy evaluation equations on non-reachable

trajectories using the standard recursion to simplify proofs, as we have done here. Formally, this is equivalent to assuming the process starts initially at τ_h instead of just conditioning on reaching τ_h, or defining the values to correspond to policy evaluation equations directly. This is consistent with the usual definition when $\mathbb{P}^\pi_M[\tau_h] > 0$ but gives it a defined value also when $\mathbb{P}^\pi_M[\tau_h] = 0$. In either case, this detail only means our recursive definition of $\mathcal{V}$ is a superset rather than exactly the set of all values as we defined in the main text. This does not effect the final results since unreachable trajectories do not effect π's overall value in the MDP anyway, and only effects the interpretations of some intermediate variables.

D.2.2 Proof of Proposition 15

Proof. By definition of V^*_M and C^*_M,

$$\begin{aligned} V^*_M > -\infty &\iff \exists \pi \in \Pi^D,\ C^\pi_M \leq B \wedge V^\pi_M \geq V^*_M \\ &\iff C^*_M \leq B. \end{aligned}$$

For the second claim, we observe that if $V^*_M > -\infty$ then by the above argument any optimal deterministic policy π for COVER satisfies $C^\pi_M = C^*_M \leq B$ and $V^\pi_M \geq V^*_M$. Thus, $COVER \subseteq PACK$. □

D.2.3 Proof of Lemma 21

Proof. We proceed by induction on h. Let $s \in \mathcal{S}$ be arbitrary.

Base Case. For the base case, we consider $h = H + 1$. In this case, we know that for any $\pi \in \Pi^D$ and any $\tau \in \mathcal{H}_{H+1}$, $V^\pi_{H+1}(\tau_{H+1}) = 0 \in \{0\} = \mathcal{V}_{H+1}(s)$ by definition. Furthermore, $|\mathcal{V}_{H+1}(s)| = 1 = A^0 = A^{\Sigma_{t=H+1}^{H} S^t}$.

Inductive Step. For the inductive step, we consider $h \leq H$. In this case, we know that for any $\pi \in \Pi^D$ and any $\tau_h \in \mathcal{H}_h$, if $s = s_h(\tau_h)$ and $a = \pi_h(\tau_h)$, then the policy evaluation equations imply,

$$V_h^\pi(\tau_h) = r_h(s,a) + \sum_{s'} P_h(s' \mid s,a) V_{h+1}^\pi(\tau_h, a, s').$$

We know by the induction hypothesis that $V_{h+1}^\pi(\tau_h, a, s') \in \mathcal{V}_{h+1}(s')$. Thus, by (D.6), $V_h^\pi(\tau_h) \in \mathcal{V}_h(s)$. Lastly, we see by (D.6) and the induction hypothesis that,

$$|\mathcal{V}_h(s)| \leq A \prod_{s'} |\mathcal{V}_{h+1}(s')| \leq A \prod_{s'} A^{\sum_{t=h+1}^H S^{H-t}} = A^{1+S\sum_{t=h+1}^H S^{H-t}} = A^{\sum_{t=h}^H S^{H-t}}.$$

This completes the proof. □

D.2.4 Proof of Lemma 22

Proof. We proceed by induction on h. Let $\tau_h \in \mathcal{H}_h$ and $v \in \mathcal{V}$ be arbitrary and suppose that $s = s_h(\tau_h)$. We let $C_h^*(\tau_h, v)$ denote the minimum for the RHS of (D.8).

Base Case. For the base case, we consider $h = H + 1$. Observe that for any $\pi \in \Pi^D$, $V_{H+1}^\pi(\tau_{H+1}) = 0$ by definition. Thus, there exists a $\pi \in \Pi^D$ satisfying $V_{H+1}^\pi(\tau_{H+1}) \geq v$ if and only if $v \leq 0$. We also know by definition that any such policy π satisfies $C_{H+1}^\pi(\tau_{H+1}) = 0$ and if no such policy exists $C_{H+1}^*(\tau_{H+1}, v) = \infty$ by convention. Therefore, we see that $C_{H+1}^*(\tau_{H+1}, v) = \chi_{\{v \leq 0\}}$. Then, by definition of the base case for $\bar{C}$, it follows that,

$$\bar{C}_{H+1}^*(s, v) = \chi_{\{v \leq 0\}} = C_{H+1}^*(\tau_{H+1}, v).$$

Inductive Step. For the inductive step, we consider $h \leq H$. If $C_h^*(\tau_h, v) = \infty$, then trivially $\bar{C}_h^*(s, v) \leq C_h^*(\tau_h, v)$. Instead, suppose that $C_h^*(\tau_h, v) < \infty$. Then, there must exist a feasible $\pi \in \Pi^D$ satisfying $V_h^\pi(\tau_h) \geq v$. Let $a^* = \pi_h(\tau_h)$. By the policy evaluation equations, we know that,

$$V_h^\pi(\tau_h) = r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a^*) V_{h+1}^\pi(\tau_h, a^*, s').$$

For each $s' \in \mathcal{S}$, define $v_{s'}^* \stackrel{\text{def}}{=} V_{h+1}^\pi(\tau_h, a^*, s')$ and observe that $v_{s'}^* \in \mathcal{V}_{h+1}(s') \subseteq \mathcal{V}$ by Lemma 21. Thus, we see that $(a^*, \mathbf{v}^*) \in \mathcal{A} \times \mathcal{V}^S$ and $r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) v_{s'} \geq v$, which implies $(a^*, \mathbf{v}^*) \in \bar{\mathcal{A}}_h(s, v)$.

Since π satisfies $V_{h+1}^\pi(\tau_h, a^*, s') \geq v_{s'}^*$, it is clear that $C_{h+1}^*(s', v_{s'}^*) \leq C_{h+1}^\pi(\tau_h, a^*, s')$. The induction hypothesis implies that $\bar{C}_{h+1}^*(s', v_{s'}^*) \leq C_{h+1}^*(s', v_{s'}^*) \leq C_{h+1}^\pi(\tau_h, a^*, s')$. The optimality equations for $\bar{M}$ then imply that,

$$\begin{aligned}
\bar{C}_h^*(s, v) &= \min_{(a, \mathbf{v}) \in \bar{\mathcal{A}}_h(s, v)} c_h(s, a) + f\left(\left(P_h(s' \mid s, a), \bar{C}_{h+1}^*(s', v_{s'})\right)_{s' \in P_h(s, a)}\right) \\
&\leq c_h(s, a^*) + f\left(\left(P_h(s' \mid s, a^*), \bar{C}_{h+1}^*(s', v_{s'}^*)\right)_{s' \in P_h(s, a^*)}\right) \\
&\leq c_h(s, a^*) + f\left(\left(P_h(s' \mid s, a), C_{h+1}^\pi(\tau_h, a^*, s')\right)_{s' \in P_h(s, a^*)}\right) \\
&= C_h^\pi(\tau_h).
\end{aligned}$$

The first inequality used the fact that $(a^*, \mathbf{v}^*) \in \bar{\mathcal{A}}_h(s, v)$. The second inequality relied on f being non-decreasing and the induction hypothesis. The final equality used (TR).

Since π was an arbitrary feasible policy for the optimization defining $C_h^*(\tau_h, v)$, we see that $\bar{C}_h^*(s, v) \leq C_h^*(\tau_h, v)$. This completes the proof. □

D.2.5 Proof of Lemma 23

Proof. We proceed by induction on h. Let $(s, v) \in \bar{\mathcal{S}}$ be arbitrary.

Base Case. For the base case, we consider $h = H+1$. By definition and assumption, $\bar{C}^{\pi}_{H+1}(s, v) = \chi_{\{v \leq 0\}} < \infty$. Thus, it must be the case that $v \leq 0$ and so by definition $\bar{V}^{\pi}_{H+1}(s, v) = 0 \geq v$.

Inductive Step. For the inductive step, we consider $h \leq H$. We decompose $\pi_h(s, v) = (a, \mathbf{v})$ where we know $(a, \mathbf{v}) \in \bar{\mathcal{A}}_h(s, v)$ since π has finite cost[2]. Moreover, it must be the case that for any $s' \in \mathcal{S}$ with $P_h(s' \mid s, a) > 0$ that $\bar{C}^{\pi}_{h+1}(s', v_{s'}) < \infty$ otherwise the property that f outputs ∞ when inputted an ∞ would imply a contradiction:

$$\begin{aligned}
\bar{C}^{\pi}_h(s, v) &= c_h(s, a) + f\left(\left(P_h(s' \mid s, a), \bar{C}^{\pi}_{h+1}\left(s', v_{s'}\right)\right)_{s' \in P_h(s,a)}\right) \\
&= c_h(s, a) + f(\ldots, \infty, \ldots) \\
&= \infty.
\end{aligned}$$

Thus, the induction hypothesis implies that $\bar{V}^{\pi}_{h+1}(s', v_{s'}) \geq v_{s'}$ for any such $s' \in \mathcal{S}$. By the policy evaluation equations, we see that,

$$\begin{aligned}
\bar{V}^{\pi}_h(s, v) &= r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) \bar{V}^{\pi}_{h+1}(s', v_{s'}) \\
&\geq r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) v_{s'} \\
&\geq v.
\end{aligned}$$

The third line uses the definition of $\bar{\mathcal{A}}_h(s, v)$. This completes the proof. □

[2] By convention, we assume $\min \varnothing = \infty$

D.2.6 Proof of Theorem 11

Proof. If $\bar{C}_1^*(s_0, v) > B$ for all $v \in \mathcal{V}$, then $C_M^* > B$ since otherwise we would have $\bar{C}_1^*(s_0, v) \leq C_1^*(s_0, v) = C_M^* \leq B$ by Lemma 22. Thus, if Algorithm 9 outputs "infeasible" it is correct.

On the other hand, suppose that there exists some $v \in \mathcal{V}$ for which $\bar{C}_1^*(s_0, v) \leq B$. By standard MDP theory, we know that since $\pi \in \Pi^D$ is a solution to $\bar{M}$, it must satisfy the optimality equations. In particular, $\bar{C}_1^\pi(s_0, v) = \bar{C}_1^*(s_0, v) \leq B$. Since $C_M^\pi = \bar{C}_1^\pi(s_0, v)$[3], we see that there exists a $\pi \in \Pi^D$ for which $C_M^\pi \leq B$ and so $V_M^* > -\infty$.

Since V_M^* is the value of some deterministic policy, Lemma 21 implies that $V_M^* \in \mathcal{V}$. Thus, Lemma 23 implies that $V_1^\pi(s_0, V_M^*) \geq V_M^*$ and $C_1^\pi(s_0, V_M^*) \leq C_1^*(s_0, V_M^*) \leq B$. Consequently, running π with initial state $\bar{s}_0 = (s_0, V_M^*)$ is an optimal solution to (CON). In either case, Algorithm 9 is correct. □

D.3 Proofs for Section 5.4

Definition 28. We define the exact partial sum,

$$\sigma_{h,\mathbf{v}}^{s,a}(t, u) \stackrel{\text{def}}{=} u + \sum_{s'=t}^{S} P_h(s' \mid s, a) v_{s'}. \tag{D.9}$$

Observation 7. *We observe that both σ and $\hat{\sigma}$ can be computed recursively. Specifically, $\sigma_{h,\mathbf{v}}^{s,a}(S+1, u) = u$ and $\sigma_{h,\mathbf{v}}^{s,a}(t, u) = \sigma_{h,\mathbf{v}}^{s,a}(t, u + P_h(t \mid s, a) v_t)$. Similarly, $\hat{\sigma}_{h,\mathbf{v}}^{s,a}(S+1, u) = u$ and $\hat{\sigma}_{h,\mathbf{v}}^{s,a}(t, u) = \sigma_{h,\mathbf{v}}^{s,a}(t, \lfloor u + P_h(t \mid s, a) v_t \rfloor_{\mathcal{G}})$.*

For completeness, and to assist with other arguments, we also prove the exact

[3] We can view $\bar{C}$ ($\bar{V}$) as the extension of C (V) needed to formally evaluate memory-augmented policies. Since we consider deterministic policies, it is trivial to convert any memory-augmented policy into a history-dependent policy that is defined in the original environment M.

recursion we presented in Definition 13 is correct using Lemma 24.

Lemma 24. *For any $t \in [S+1]$ and $u \in \mathbb{R}$, we have that,*

$$g_{h,v}^{s,a}(t,u) = \min_{\mathbf{v} \in \mathcal{V}^{S-t+1}} g_{h,\mathbf{v}}^{s,a}(t) \quad s.t. \quad u + \sum_{s'=t}^{S} P_h(s' \mid s,a) v_{s'} \geq v. \tag{D.10}$$

Moreover, $\bar{C}_h^*(s,v) = \min_{a \in \mathcal{A}} c_h(s,a) + g_{h,v}^{s,a}(1, r_h(s,a))$.

D.3.1 Proof of Lemma 24

Proof. We proceed by induction on t.

Base Case. For the base case, we consider $t = S+1$. Since $\sum_{s'=S+1}^{S} P_h(s' \mid s,a) v_{s'} = 0$ is the empty sum, the condition $u + \sum_{s'=S+1}^{S} P_h(s' \mid s,a) v_{s'} \geq v$ is true iff $u \geq v$. Also, for any $\mathbf{v}$, $g_{h,\mathbf{v}}^{s,a}(S+1) = 0$ by definition. Thus, the minimum defining $g_{h,\mathbf{v}}^{s,a}(S+1,u)$ is 0 when $u \geq v$ and is ∞ due to infeasibility otherwise. In symbols, $g_{h,v}^{s,a}(S+1,u) = \chi_{\{u \geq v\}}$ as was to be shown.

Inductive Step. For the inductive step, we consider $t \leq S$. We see that, $g^{s,a}_{h,v}(t,u) =$

$$
\begin{aligned}
&\min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t+1} \\ u+\sum_{s'=t}^{S} P_h(s'|s,a)v_{s'}\geq v}} g^{s,a}_{h,\mathbf{v}}(t) \\
&= \min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t+1} \\ u+\sum_{s'=t}^{S} P_h(s'|s,a)v_{s'}\geq v}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), g^{s,a}_{h,\mathbf{v}}(t+1)\right) \\
&= \min_{v_t\in\mathcal{V}} \min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t} \\ (u+P_h(t|s,a)v_t)+\sum_{s'=t+1}^{S} P_h(s'|s,a)v_{s'}\geq v}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), g^{s,a}_{h,\mathbf{v}}(t+1)\right) \\
&= \min_{v_t\in\mathcal{V}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), \min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t} \\ (u+P_h(t|s,a)v_t)+\sum_{s'=t+1}^{S} P_h(s'|s,a)v_{s'}\geq v}} g^{s,a}_{h,\mathbf{v}}(t+1)\right) \\
&= \min_{v_t\in\mathcal{V}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), g^{s,a}_{h,v}(t+1, u+P_h(t \mid s,a)v_t)\right)
\end{aligned}
$$

The second lined used (SR). The third line split the optimization into the first decision and the remaining decisions and decomposed the sum in the constraint. The fourth line used the fact that α is a non-decreasing function of both its arguments and the fact that the second optimization only concerns the second argument. The last line used the induction hypothesis.

The observation that $\min_{a\in\mathcal{A}} c_h(s,a) + g^{s,a}_{h,v}(1, r_h(s,a)) = \bar{C}^*_h(s,v)$ then follows from the definition of $\bar{\mathcal{A}}_h(s,v)$ and (BU):

$$
\begin{aligned}
\min_{a\in\mathcal{A}} c_h(s,a) + g^{s,a}_{h,v}(1, r_h(s,a)) &= \min_{a\in\mathcal{A}} c_h(s,a) + \min_{\substack{\mathbf{v}\in\mathcal{V}^{S} \\ r_h(s,a)+\sum_{s'} P_h(s'|s,a)v_s\geq v}} g^{s,a}_{h,\mathbf{v}}(1) \\
&= \min_{a\in\mathcal{A}} \min_{\substack{\mathbf{v}\in\mathcal{V}^{S} \\ r_h(s,a)+\sum_{s'} P_h(s'|s,a)v_s\geq v}} c_h(s,a) + g^{s,a}_{h,\mathbf{v}}(1) \\
&= \min_{(a,\mathbf{v})\in\bar{\mathcal{A}}_h(s,v)} c_h(s,a) + g^{s,a}_{h,\mathbf{v}}(1) \\
&= \bar{C}^*_h(s,v).
\end{aligned}
$$

□

D.3.2 Proof of Lemma 11

Proof. We proceed by induction on t.

Base Case. For the base case, we consider $t = S+1$. By definition, $\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(S+1,u) = u$ so the constraint is satisfied iff $u \geq v$. Since for any $\hat{\mathbf{v}}$, $\hat{g}^{s,a}_{h,\hat{\mathbf{v}}}(S+1) = 0$ by definition, the minimum defining $\hat{g}^{s,a}_{h,\hat{\mathbf{v}}}(S+1,u)$ is 0 when $u \geq v$ and is ∞ due to infeasibility otherwise. In symbols, $\hat{g}^{s,a}_{h,v}(S+1,u) = \chi_{\{u \geq v\}}$ as was to be shown.

Inductive Step. For the inductive step, we consider $t \leq S$. We see that,

$$
\begin{aligned}
\hat{g}^{s,a}_{h,v}(t,u) &= \min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t+1} \\ \hat{\sigma}^{s,a}_{h,\mathbf{v}}(t,u)\geq v}} g^{s,a}_{h,\mathbf{v}}(t) \\
&= \min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t+1} \\ \hat{\sigma}^{s,a}_{h,\mathbf{v}}(t,u)\geq v}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), g^{s,a}_{h,\mathbf{v}}(t+1)\right) \\
&= \min_{v_t\in\mathcal{V}} \min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t} \\ \hat{\sigma}^{s,a}_{h,\mathbf{v}}(t+1,\lfloor u+P_h(t|s,a)v_t\rfloor_{\mathcal{G}})\geq v}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), g^{s,a}_{h,\mathbf{v}}(t+1)\right) \\
&= \min_{v_t\in\mathcal{V}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), \min_{\substack{\mathbf{v}\in\mathcal{V}^{S-t} \\ \hat{\sigma}^{s,a}_{h,\mathbf{v}}(t+1,\lfloor u+P_h(t|s,a)v_t\rfloor_{\mathcal{G}})\geq v}} g^{s,a}_{h,\mathbf{v}}(t+1)\right) \\
&= \min_{v_t\in\mathcal{V}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), \hat{g}^{s,a}_{h,v}(t+1, \lfloor u+P_h(t \mid s,a)v_t\rfloor_{\mathcal{G}})\right)
\end{aligned}
$$

The second lined used (SR). The third line split the optimization into the first decision and the remaining decisions and used the recursive definition of $\hat{\sigma}$ in the constraint. The fourth line used the fact that α is a non-decreasing function of both its arguments and the fact that the second optimization only concerns the second argument. The last line used the induction hypothesis. □

D.3.3 Proof of Proposition 16

Proof. The runtime guarantee is easily seen since Algorithm 12 consists of nested loops. The fact that it computes an optimal solution for $\bar{M}$ absent rounding or lower bounding follows immediately from Lemma 24. □

D.4 Proofs for Section 5.5

D.4.1 Helpful Technical Lemmas (Additive)

The following claims all assume Definition 17.

Observation 8. *For any $v \in \mathbb{R}$,*

$$v - \delta \leq \lfloor v \rfloor_{\mathcal{G}} \leq v. \tag{D.11}$$

Lemma 25. *For any $h \in [H]$, $s \in \mathcal{S}$, $a \in \mathcal{A}$, $\mathbf{v} \in \mathbb{R}^S$, $u \in \mathbb{R}$, and $t \in [S+1]$, we have,*

$$\sigma^{s,a}_{h,\mathbf{v}}(t,u) - (S-t+1)\delta \leq \hat{\sigma}^{s,a}_{h,\mathbf{v}}(t,u) \leq \sigma^{s,a}_{h,\mathbf{v}}(t,u). \tag{D.12}$$

Lemma 26 (Cost)**.** *For any $h \in [H+1]$ and $(s,v) \in \bar{\mathcal{S}}$, $\hat{C}^*_h(s, \lfloor v \rfloor_{\mathcal{G}}) \leq \bar{C}^*_h(s,v)$.*

Lemma 27 (Approximation)**.** *Suppose that $\pi \in \Pi^D$. For all $h \in [H+1]$ and $(s,\hat{v}) \in \hat{\mathcal{S}}$, if $\hat{C}^\pi_h(s,\hat{v}) < \infty$, then $\hat{V}^\pi_h(s,\hat{v}) \geq \hat{v} - \delta(S+1)(H-h+1)$.*

D.4.2 Helpful Technical Lemmas (Relative)

The following claims all assume Definition 18.

Observation 9. *For any $v \in \mathbb{R}$,*

$$v(1-\delta) \leq \lfloor v \rfloor_{\mathcal{G}} \leq v. \tag{D.13}$$

Lemma 28. *For any $h \in [H]$, $s \in \mathcal{S}$, $a \in \mathcal{A}$, $\mathbf{v} \in \mathbb{R}^S_{\geq 0}$, $u \in \mathbb{R}_{\geq 0}$, and $t \in [S+1]$, we have,*

$$\sigma^{s,a}_{h,\mathbf{v}}(t,u)(1-\delta)^{S-t+1} \leq \hat{\sigma}^{s,a}_{h,\mathbf{v}}(t,u) \leq \sigma^{s,a}_{h,\mathbf{v}}(t,u). \tag{D.14}$$

Lemma 29 (Cost)**.** *Suppose all rewards are non-negative. For any $h \in [H+1]$ and $(s,v) \in \bar{\mathcal{S}}$, $\hat{C}^*_h(s, \lfloor v \rfloor_{\mathcal{G}}) \leq \bar{C}^*_h(s,v)$.*

Lemma 30 (Approximation)**.** *Suppose all rewards are non-negative and $\pi \in \Pi^D$. For all $h \in [H+1]$ and $(s,\hat{v}) \in \hat{\mathcal{S}}$, if $\hat{C}^\pi_h(s,\hat{v}) < \infty$, then $\hat{V}^\pi_h(s,\hat{v}) \geq \hat{v}(1-\delta)^{(S+1)(H-h+1)}$.*

D.4.3 Proof of Observation 8

Proof. Using properties of the floor function, we can infer that,

$$\lfloor v \rfloor_{\mathcal{G}} = \left\lfloor \frac{v}{\delta} \right\rfloor \delta \leq \frac{v}{\delta}\delta = v,$$

and,

$$\lfloor v \rfloor_{\mathcal{G}} = \left\lfloor \frac{v}{\delta} \right\rfloor \delta \geq (\left\lceil \frac{v}{\delta} \right\rceil - 1)\delta = \left\lceil \frac{v}{\delta} \right\rceil \delta - \delta \geq v - \delta.$$

□

D.4.4 Proof of Lemma 25

Proof. We proceed by induction on t.

Base Case. For the base case, we consider $t = S+1$. By definition, we have $\hat{\sigma}^{s,a}_{h,\mathbf{v}}(S+1,u) = u = \sigma^{s,a}_{h,\mathbf{v}}(S+1,u)$.

Inductive Step. For the inductive step, we consider $t \leq S$. We first see that,

$$\begin{aligned}
\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(t,u) &= \hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}) \\
&\leq \sigma^{s,a}_{h,\hat{\mathbf{v}}}(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}) \\
&= \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}} + \sum_{s'=t+1}^{S} P_h(s' \mid s,a)\hat{v}_t \\
&\leq u + \sum_{s'=t}^{S} P_h(s' \mid s,a)\hat{v}_t \\
&= \sigma^{s,a}_{h,\hat{\mathbf{v}}}(t,u).
\end{aligned}$$

The first inequality used the induction hypothesis and the second inequality used the fact that $\lfloor x \rfloor_{\mathcal{G}} \leq x$.

We also see that,

$$\begin{aligned}
\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(t,u) &= \hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}) \\
&\geq \sigma^{s,a}_{h,\hat{\mathbf{v}}}(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}) - \delta(S-t) \\
&= \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}} + \sum_{s'=t+1}^{S} P_h(s' \mid s,a)\hat{v}_t - \delta(S-t) \\
&\geq u + \sum_{s'=t}^{S} P_h(s' \mid s,a)\hat{v}_t - \delta(S-t+1) \\
&= \sigma^{s,a}_{h,\hat{\mathbf{v}}}(t,u) - \delta(S-t+1).
\end{aligned}$$

The first inequality used the induction hypothesis and the second inequality used the fact that $\lfloor x \rfloor_{\mathcal{G}} \geq x - \delta$. □

D.4.5 Proof of Lemma 26

Proof. We proceed by induction on h. Let $(s,v) \in \bar{\mathcal{S}}$ be arbitrary.

Base Case. For the base case, we consider $h = H + 1$. Since $\lfloor v \rfloor_{\mathcal{G}} \leq v$, we immediately see,

$$\hat{C}^*_{H+1}(s, \lfloor v \rfloor_{\mathcal{G}}) = \chi_{\left\{\lfloor v \rfloor_{\mathcal{G}} \leq 0\right\}} \leq \chi_{\{v \leq 0\}} = \bar{C}^*_{H+1}(s, v).$$

Inductive Step. For the inductive step, we consider $h \leq H$. If $\bar{C}^*_h(s, v) = \infty$, then trivially $\hat{C}^*_h(s, \lfloor v \rfloor_{\mathcal{G}}) \leq \bar{C}^*_h(s, v)$. Instead, suppose that $\bar{C}^*_h(s, v) < \infty$. Let π be a solution to the optimality equations for $\bar{M}$ so that $\bar{C}^\pi_h(s, v) = \bar{C}^*_h(s, v) < \infty$. Since $\bar{C}^*_h(s, v) < \infty$, we know that $(a^*, \mathbf{v}^*) = \pi_h(s, v) \in \bar{\mathcal{A}}_h(s, v)$. By the definition of $\bar{\mathcal{A}}_h(s, v)$, we know that,

$$\sigma^{s,a^*}_{h,\mathbf{v}^*}(1, r_h(s, a^*)) = r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a^*) v^*_{s'} \geq v \geq \lfloor v \rfloor_{\mathcal{G}}.$$

For each $s' \in \mathcal{S}$, define $\hat{v}^*_{s'} \stackrel{\text{def}}{=} \lfloor v^*_{s'} \rfloor_{\mathcal{G}}$ and recall that $v^*_{s'} \in \mathcal{V}$. We first observe that,

$$\begin{aligned}
\sigma^{s,a^*}_{h,\hat{\mathbf{v}}^*}(1, r_h(s, a^*)) &= r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a) \lfloor v_{s'} \rfloor_{\mathcal{G}} \\
&\geq r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a)(v_{s'} - \delta) \\
&= r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a) v_{s'} - \delta \\
&= \sigma^{s,a^*}_{h,\mathbf{v}^*}(1, r_h(s, a^*)) - \delta.
\end{aligned}$$

Then by Lemma 25,

$$\begin{aligned}
\hat{\sigma}^{s,a^*}_{h,\hat{\mathbf{v}}^*}(1, r_h(s,a^*)) &\geq \sigma^{s,a^*}_{h,\hat{\mathbf{v}}^*}(1, r_h(s,a^*)) - \delta S \\
&\geq \sigma^{s,a^*}_{h,\mathbf{v}^*}(1, r_h(s,a^*)) - \delta(S+1) \\
&\geq \lfloor v \rfloor_{\mathcal{G}} - \delta(S+1) \\
&= \kappa(\lfloor v \rfloor_{\mathcal{G}}).
\end{aligned}$$

Thus, $(a^*, \hat{\mathbf{v}}^*) \in \hat{\mathcal{A}}_h(s, \lfloor v \rfloor_{\mathcal{G}})$.

Since $v^*_{s'} \in \mathcal{V}$, the induction hypothesis implies that $\hat{C}^*_{h+1}(s', \hat{v}^*_{s'}) \leq \bar{C}^*_{h+1}(s', v^*_{s'}) = \bar{C}^{\pi}_{h+1}(s', v^*_{s'})$. The optimality equations for $\hat{M}$ then imply that,

$$\begin{aligned}
\hat{C}^*_h(s, \lfloor v \rfloor_{\mathcal{G}}) &= \min_{(a,\hat{\mathbf{v}}) \in \hat{\mathcal{A}}_h(s,v)} c_h(s,a) + f\left(\left(P_h(s' \mid s,a), \hat{C}^*_{h+1}(s', \hat{v}_{s'})\right)_{s' \in P_h(s,a)}\right) \\
&\leq c_h(s,a^*) + f\left(\left(P_h(s' \mid s,a^*), \hat{C}^*_{h+1}(s', \hat{v}^*_{s'})\right)_{s' \in P_h(s,a^*)}\right) \\
&\leq c_h(s,a^*) + f\left(\left(P_h(s' \mid s,a), \bar{C}^{\pi}_{h+1}(s', v^*_{s'})\right)_{s' \in P_h(s,a^*)}\right) \\
&= \bar{C}^{\pi}_h(s,v) \\
&= \bar{C}^*_h(s,v).
\end{aligned}$$

The first inequality used the fact that $(a^*, \mathbf{v}^*) \in \hat{\mathcal{A}}_h(s,v)$. The second inequality relied on f being non-decreasing and the induction hypothesis. The penultimate equality used (TR). This completes the proof. □

D.4.6 Proof of Lemma 27

Proof. We proceed by induction on h. Let $(s, \hat{v}) \in \hat{\mathcal{S}}$ be arbitrary.

Base Case. For the base case, we consider $h = H+1$. By definition and assumption, $\hat{C}^{\pi}_{H+1}(s, \hat{v}) = \chi_{\{\hat{v} \leq 0\}} < \infty$. Thus, it must be the case that $\hat{v} \leq 0$ and so by definition $\hat{V}^{\pi}_{H+1}(s, \hat{v}) = 0 \geq \hat{v}$.

Inductive Step. For the inductive step, we consider $h \leq H$. As in the proof of Lemma 23, we know that $\pi_h(s, v) = (a, \hat{\mathbf{v}}) \in \hat{\mathcal{A}}_h(s, \hat{v})$ and for any $s' \in \mathcal{S}$ with $P_h(s' \mid s, a) > 0$ that $\hat{C}^{\pi}_{h+1}(s', v_{s'}) < \infty$. Thus, the induction hypothesis implies that $\hat{V}^{\pi}_{h+1}(s', \hat{v}_{s'}) \geq \hat{v}_{s'} - \delta(S+1)(H-h)$ for any such $s' \in \mathcal{S}$. By the policy evaluation equations, we see that,

$$
\begin{aligned}
\hat{V}^{\pi}_{h}(s, \hat{v}) &= r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) \hat{V}^{\pi}_{h+1}(s', \hat{v}_{s'}) \\
&\geq r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) \hat{v}_{s'} - \delta(S+1)(H-h) \\
&= \sigma^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s, a)) - \delta(S+1)(H-h) \\
&\geq \hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s, a)) - \delta(S+1)(H-h) \\
&\geq \hat{v} - \delta(S+1) - \delta(S+1)(H-h) \\
&= \hat{v} - \delta(S+1)(H-h+1).
\end{aligned}
$$

The first inequality used the induction hypothesis. The second inequality used Lemma 25. The third inequality used the fact that by definition of $\hat{\mathcal{A}}_h(s, \hat{v})$ and κ, $\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s, a)) \geq \kappa(\hat{v}) = \hat{v} - \delta(S+1)$. This completes the proof. □

D.4.7 Proof of Theorem 12

Proof.

Correctness. If $\hat{C}_1^*(s_0, v) > B$ for all $\hat{v} \in \hat{\mathcal{V}}$, then $C_M^* > B$ since otherwise we would have $\hat{C}_1^*(s_0, \lfloor v \rfloor_{\mathcal{G}}) \leq \bar{C}_1^*(s_0, v) \leq C_M^* \leq B$ by Lemma 26. Thus, if Algorithm 13 outputs "infeasible" it is correct.

On the other hand, suppose that there exists some $\hat{v} \in \hat{\mathcal{V}}$ for which $\hat{C}_1^*(s_0, \hat{v}) \leq B$. By standard MDP theory, we know that since $\pi \in \Pi^D$ is a solution to $\hat{M}$, it must satisfy the optimality equations. In particular, $\hat{C}_1^\pi(s_0, \hat{v}) = \hat{C}_1^*(s_0, v) \leq B$. As in the proof of Theorem 11, since $C_M^\pi = \hat{C}_1^\pi(s_0, \hat{v})$, we see that there exists a $\pi \in \Pi^D$ for which $C_M^\pi \leq B$ and so $V_M^* > -\infty$.

Since V_M^* is the value of some deterministic policy, Lemma 21 implies that $V_M^* \in \mathcal{V}$. Thus, Lemma 27 implies that $\hat{V}_1^\pi(s_0, \lfloor V_M^* \rfloor_{\mathcal{G}}) \geq \lfloor V_M^* \rfloor_{\mathcal{G}} - \delta(S+1)H \geq V_M^* - \delta(1 + (S+1)H) = V_M^* - \epsilon$ and $\hat{C}_1^\pi(s_0, V_M^*) \leq C_1^*(s_0, V_M^*) \leq B$. Consequently, running π with initial state $\bar{s}_0 = (s_0, \lfloor V_M^* \rfloor_{\mathcal{G}})$ is an optimal solution to (CON). In either case, Algorithm 13 is correct.

Complexity. For the complexity claim, we observe that the running time of Algorithm 13 is $O(HS^2A|\hat{\mathcal{V}}|^2|\hat{\mathcal{U}}|)$. To bound $|\hat{\mathcal{V}}|$, we observe that the number of integer multiples of δ required to capture the range $[-Hr_{max}, Hr_{max}]$ is at most $O(\frac{Hr_{max}}{\delta}) = O(H^2 S r_{max}/\epsilon)$ by definition of δ. Moreover, $|\hat{\mathcal{U}}| = O(|\hat{\mathcal{V}}| + S) = O(|\hat{\mathcal{V}}|)$ for sufficiently large $\frac{r_{max}}{\epsilon}$.

In particular, we see that the range of the rounded sums defining $\hat{\mathcal{U}}$ is at widest $[-2Hr_{max} - \delta S, 2Hr_{max}]$ since for any $t+1$ the rounded input is,

$$\left\lfloor \lfloor r_h(s,a) + P_h(1 \mid s,a)\hat{\mathbf{v}}_1 \rfloor_{\mathcal{G}} + \ldots + P_h(t \mid s,a)\hat{\mathbf{v}}_t \right\rfloor_{\mathcal{G}} \leq r_h(s,a) + \sum_{s'=1}^{t} P_h(s' \mid s,a)\hat{\mathbf{v}}_{s'},$$

which is at most $2Hr_{max}$, and,

$$\begin{aligned}\Big\lfloor \lfloor r_h(s,a) + P_h(1 \mid s,a)\hat{\mathbf{v}}_1 \rfloor_{\mathcal{G}} + \ldots + P_h(t \mid s,a)\hat{\mathbf{v}}_t \Big\rfloor_{\mathcal{G}} \\ \geq r_h(s,a) + \sum_{s'=1}^{t} P_h(s' \mid s,a)\hat{\mathbf{v}}_{s'} - \delta t,\end{aligned}$$

which is at least $-2Hr_{max} - \delta S$. Overall, we see that $O(|\hat{\mathcal{V}}|^2|\hat{\mathcal{U}}|) = O(|\hat{\mathcal{V}}|^3) = O(H^6S^3r_{max}^3/\epsilon^3)$ implying that the total run time is $O(H^7S^5Ar_{max}^3/\epsilon^3)$ as claimed.

□

D.4.8 Proof of Observation 9

Proof. Using properties of the floor function, we can infer that,

$$\lfloor v \rfloor_{\mathcal{G}} = v^{min}\left(\frac{1}{1-\delta}\right)^{\left\lfloor \log_{\frac{1}{1-\delta}} \frac{v}{v^{min}} \right\rfloor} \leq v^{min}\left(\frac{1}{1-\delta}\right)^{\log_{\frac{1}{1-\delta}} \frac{v}{v^{min}}} = \frac{v}{v^{min}}v^{min} = v,$$

and,

$$\lfloor v \rfloor_{\mathcal{G}} = v^{min}\left(\frac{1}{1-\delta}\right)^{\left\lfloor \log_{\frac{1}{1-\delta}} \frac{v}{v^{min}} \right\rfloor} \geq v^{min}\left(\frac{1}{1-\delta}\right)^{\log_{\frac{1}{1-\delta}} \frac{v}{v^{min}} - 1} = v(1-\delta).$$

□

D.4.9 Proof of Lemma 28

Proof. We proceed by induction on t.

Base Case. For the base case, we consider $t = S + 1$. By definition, we have $\hat{\sigma}_{h,\mathbf{v}}^{s,a}(S+1,u) = u = \sigma_{h,\mathbf{v}}^{s,a}(S+1,u)$.

Inductive Step. For the inductive step, we consider $t \leq S$. We first see that,

$$\begin{aligned}
\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(t,u) &= \hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}) \\
&\leq \sigma^{s,a}_{h,\hat{\mathbf{v}}}(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}) \\
&= \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}} + \sum_{s'=t+1}^{S} P_h(s' \mid s,a)\hat{v}_t \\
&\leq u + \sum_{s'=t}^{S} P_h(s' \mid s,a)\hat{v}_t \\
&= \sigma^{s,a}_{h,\hat{\mathbf{v}}}(t,u).
\end{aligned}$$

The first inequality used the induction hypothesis and the second inequality used the fact that $\lfloor x \rfloor_{\mathcal{G}} \leq x$.

We also see that,

$$\begin{aligned}
\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(t,u) &= \hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}\left(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}\right) \\
&\geq \sigma^{s,a}_{h,\hat{\mathbf{v}}}\left(t+1, \lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}}\right)(1-\delta)^{S-t} \\
&= \left(\lfloor u + P_h(t \mid s,a)\hat{v}_t \rfloor_{\mathcal{G}} + \sum_{s'=t+1}^{S} P_h(s' \mid s,a)\hat{v}_t\right)(1-\delta)^{S-t} \\
&\geq \left((1-\delta)u + (1-\delta)\sum_{s'=t}^{S} P_h(s' \mid s,a)\hat{v}_t\right)(1-\delta)^{S-t} \\
&= \sigma^{s,a}_{h,\hat{\mathbf{v}}}(t,u)(1-\delta)^{S-t+1}.
\end{aligned}$$

The first inequality used the induction hypothesis and the second inequality used the fact that $\lfloor x \rfloor_{\mathcal{G}} \geq x - \delta$ and the fact that all rewards and values are non-negative allowing us to add a $(1-\delta)$-factor to the other value demands. □

D.4.10 Proof of Lemma 29

Proof. We proceed by induction on h. Let $(s, v) \in \bar{\mathcal{S}}$ be arbitrary.

Base Case. For the base case, we consider $h = H + 1$. Since $\lfloor v \rfloor_{\mathcal{G}} \leq v$, we immediately see,

$$\hat{C}^*_{H+1}(s, \lfloor v \rfloor_{\mathcal{G}}) = \chi_{\left\{\lfloor v \rfloor_{\mathcal{G}} \leq 0\right\}} \leq \chi_{\{v \leq 0\}} = \bar{C}^*_{H+1}(s, v).$$

Inductive Step. For the inductive step, we consider $h \leq H$. If $\bar{C}^*_h(s, v) = \infty$, then trivially $\hat{C}^*_h(s, \lfloor v \rfloor_{\mathcal{G}}) \leq \bar{C}^*_h(s, v)$. Instead, suppose that $\bar{C}^*_h(s, v) < \infty$. Let π be a solution to the optimality equations for $\bar{M}$ so that $\bar{C}^\pi_h(s, v) = \bar{C}^*_h(s, v) < \infty$. Since $\bar{C}^*_h(s, v) < \infty$, we know that $(a^*, \mathbf{v}^*) = \pi_h(s, v) \in \bar{\mathcal{A}}_h(s, v)$. By the definition of $\bar{\mathcal{A}}_h(s, v)$, we know that,

$$\sigma^{s,a^*}_{h,\mathbf{v}^*}(1, r_h(s, a^*)) = r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a^*) v^*_{s'} \geq v \geq \lfloor v \rfloor_{\mathcal{G}}.$$

For each $s' \in \mathcal{S}$, define $\hat{v}^*_{s'} \stackrel{\text{def}}{=} \lfloor v^*_{s'} \rfloor_{\mathcal{G}}$ and recall that $v^*_{s'} \in \mathcal{V}$. We first observe that,

$$\begin{aligned}
\sigma^{s,a^*}_{h,\hat{\mathbf{v}}^*}(1, r_h(s, a^*)) &= r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a) \lfloor v_{s'} \rfloor_{\mathcal{G}} \\
&\geq r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a) v_{s'} (1 - \delta) \\
&\geq \left(r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a) v_{s'} \right) (1 - \delta) \\
&= \sigma^{s,a^*}_{h,\mathbf{v}^*}(1, r_h(s, a^*))(1 - \delta).
\end{aligned}$$

The second inequality used the fact that all rewards are non-negative. Then by Lemma 28,

$$\begin{aligned}
\hat{\sigma}^{s,a^*}_{h,\hat{\mathbf{v}}^*}(1, r_h(s,a^*)) &\geq \sigma^{s,a^*}_{h,\hat{\mathbf{v}}^*}(1, r_h(s,a^*))(1-\delta)^S \\
&\geq \sigma^{s,a^*}_{h,\mathbf{v}^*}(1, r_h(s,a^*))(1-\delta)^{S+1} \\
&\geq \lfloor v \rfloor_{\mathcal{G}} (1-\delta)^{S+1} \\
&= \kappa(\lfloor v \rfloor_{\mathcal{G}}).
\end{aligned}$$

Thus, $(a^*, \hat{\mathbf{v}}^*) \in \hat{\mathcal{A}}_h(s, \lfloor v \rfloor_{\mathcal{G}})$.

Since $v^*_{s'} \in \mathcal{V}$, the induction hypothesis implies that $\hat{C}^*_{h+1}(s', \hat{v}^*_{s'}) \leq \bar{C}^*_{h+1}(s', v^*_{s'}) = \bar{C}^{\pi}_{h+1}(s', v^*_{s'})$. The optimality equations for $\hat{M}$ then imply that,

$$\begin{aligned}
\hat{C}^*_h(s, \lfloor v \rfloor_{\mathcal{G}}) &= \min_{(a,\hat{\mathbf{v}}) \in \hat{\mathcal{A}}_h(s,v)} c_h(s,a) + f\left(\left(P_h(s' \mid s,a), \hat{C}^*_{h+1}(s', \hat{v}_{s'})\right)_{s' \in P_h(s,a)}\right) \\
&\leq c_h(s,a^*) + f\left(\left(P_h(s' \mid s,a^*), \hat{C}^*_{h+1}(s', \hat{v}^*_{s'})\right)_{s' \in P_h(s,a^*)}\right) \\
&\leq c_h(s,a^*) + f\left(\left(P_h(s' \mid s,a), \bar{C}^{\pi}_{h+1}(s', v^*_{s'})\right)_{s' \in P_h(s,a^*)}\right) \\
&= \bar{C}^{\pi}_h(s,v) \\
&= \bar{C}^*_h(s,v).
\end{aligned}$$

The first inequality used the fact that $(a^*, \mathbf{v}^*) \in \hat{\mathcal{A}}_h(s,v)$. The second inequality relied on f being non-decreasing and the induction hypothesis. The penultimate equality used (TR).

This completes the proof. □

D.4.11 Proof of Lemma 30

Proof. We proceed by induction on h. Let $(s, \hat{v}) \in \hat{\mathcal{S}}$ be arbitrary.

Base Case. For the base case, we consider $h = H+1$. By definition and assumption, $\hat{C}^{\pi}_{H+1}(s, \hat{v}) = \chi_{\{\hat{v} \leq 0\}} < \infty$. Thus, it must be the case that $\hat{v} \leq 0$ and so by definition $\hat{V}^{\pi}_{H+1}(s, \hat{v}) = 0 \geq \hat{v}$.

Inductive Step. For the inductive step, we consider $h \leq H$. As in the proof of Lemma 23, we know that $\pi_h(s, v) = (a, \hat{\mathbf{v}}) \in \hat{\mathcal{A}}_h(s, \hat{v})$ and for any $s' \in \mathcal{S}$ with $P_h(s' \mid s, a) > 0$ that $\hat{C}^{\pi}_{h+1}(s', v_{s'}) < \infty$. Thus, the induction hypothesis implies that $\hat{V}^{\pi}_{h+1}(s', \hat{v}_{s'}) \geq \hat{v}_{s'}(1-\delta)^{(S+1)(H-h)}$ for any such $s' \in \mathcal{S}$. By the policy evaluation equations, we see that,

$$\begin{aligned}
\hat{V}^{\pi}_h(s, \hat{v}) &= r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) \hat{V}^{\pi}_{h+1}(s', \hat{v}_{s'}) \\
&\geq r_h(s, a) + \sum_{s'} P_h(s' \mid s, a) \hat{v}_{s'}(1-\delta)^{(S+1)(H-h)} \\
&\geq \sigma^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s, a))(1-\delta)^{(S+1)(H-h)} \\
&\geq \hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s, a))(1-\delta)^{(S+1)(H-h)} \\
&\geq \hat{v}(1-\delta)^{S+1}(1-\delta)^{(S+1)(H-h)} \\
&= \hat{v}(1-\delta)^{(S+1)(H-h+1)}.
\end{aligned}$$

The first inequality used the induction hypothesis. The second inequality used the fact that the rewards are non-negative. The third inequality used Lemma 28. The fourth inequality used the fact that by definition of $\hat{\mathcal{A}}_h(s, \hat{v})$ and κ, $\hat{\sigma}^{s,a}_{h,\hat{\mathbf{v}}}(1, r_h(s, a)) \geq \kappa(\hat{v}) = \hat{v}(1-\delta)^{S+1}$.

This completes the proof. □

D.4.12 Proof of Theorem 13

Proof.

Correctness. If $\hat{C}_1^*(s_0, v) > B$ for all $\hat{v} \in \hat{\mathcal{V}}$, then $C_M^* > B$ since otherwise we would have $\hat{C}_1^*(s_0, \lfloor v \rfloor_{\mathcal{G}}) \leq \bar{C}_1^*(s_0, v) \leq C_M^* \leq B$ by Lemma 29. Thus, if Algorithm 13 outputs "infeasible" it is correct.

On the other hand, suppose that there exists some $\hat{v} \in \hat{\mathcal{V}}$ for which $\hat{C}_1^*(s_0, \hat{v}) \leq B$. By standard MDP theory, we know that since $\pi \in \Pi^D$ is a solution to $\hat{M}$, it must satisfy the optimality equations. In particular, $\hat{C}_1^\pi(s_0, \hat{v}) = \hat{C}_1^*(s_0, v) \leq B$. As in the proof of Theorem 11, since $C_M^\pi = \hat{C}_1^\pi(s_0, \hat{v})$, we see that there exists a $\pi \in \Pi^D$ for which $C_M^\pi \leq B$ and so $V_M^* > -\infty$.

Since V_M^* is the value of some deterministic policy, Lemma 21 implies that $V_M^* \in \mathcal{V}$. Thus, Lemma 30 implies that $\hat{V}_1^\pi(s_0, \lfloor V_M^* \rfloor_{\mathcal{G}}) \geq \lfloor V_M^* \rfloor_{\mathcal{G}} (1-\delta)^{(S+1)H} \geq V_M^*(1-\delta)^{(S+1)H+1} = V_M^*(1 - \frac{\epsilon}{(S+1)H+1})^{(S+1)H+1} \geq V_M^*(1-\epsilon)$ and $\hat{C}_1^\pi(s_0, V_M^*) \leq C_1^*(s_0, V_M^*) \leq B$. Consequently, running π with initial state $\bar{s}_0 = (s_0, \lfloor V_M^* \rfloor_{\mathcal{G}})$ is an optimal solution to (CON). In either case, Algorithm 13 is correct.

Complexity. For the complexity claim, we observe that the running time of Algorithm 13 is $O(HS^2A|\hat{\mathcal{V}}|^2|\hat{\mathcal{U}}|)$. To bound $|\hat{\mathcal{V}}|$, we observe that the number of *vmin*-scaled powers of $1/(1-\delta)$ required to capture the range $[0, Hr_{max}]$ is at most one plus the largest power needed, which is

$$\begin{aligned} O(\log_{1/(1-\delta)}(\frac{Hr_{max}}{v_{min}})) &= O(\log(\frac{Hr_{max}}{v_{min}})/\log(1/(1-\delta))) \\ &= O(\log(\frac{Hr_{max}}{v_{min}})/\delta) \\ &= O(\log(HS\frac{Hr_{max}}{p_{min}^H r_{min}})/\epsilon) \\ &= O(H^2 S \log(\frac{r_{max}}{p_{min} r_{min}})/\epsilon), \end{aligned}$$

by definition of δ and the fact that $\log(\frac{1}{1-\delta}) = -\log(1-\delta) \geq -\log(e^{-\delta}) = \delta$. Moreover, $|\hat{\mathcal{U}}| = O(|\hat{\mathcal{V}}|)$.

We see that the range of the rounded sums is at widest $[0, 2Hr_{max}]$ since for any $t+1$ rounding non-negative sums is at least 0 and,

$$\left\lfloor \lfloor r_h(s,a) + P_h(1 \mid s,a)\hat{\mathbf{v}}_1 \rfloor_{\mathcal{G}} + \ldots + P_h(t \mid s,a)\hat{\mathbf{v}}_t \right\rfloor_{\mathcal{G}} \leq r_h(s,a) + \sum_{s'=1}^{t} P_h(s' \mid s,a)\hat{\mathbf{v}}_{s'},$$

which is at most $2Hr_{max}$. Then, the same analysis from before shows that the number of scaled powers of $1/(1-\delta)$ needed to cover this interval is $O(|\hat{\mathcal{V}}|)$. Thus, we see that $O(|\hat{\mathcal{V}}|^2|\hat{\mathcal{U}}|) = O(|\hat{\mathcal{V}}|^3) = O(H^6 S^3 \log(\frac{r_{max}}{p_{min} r_{min}})^3/\epsilon^3)$ implying that the total run time is $O(H^7 S^5 A \log(\frac{r_{max}}{p_{min} r_{min}})^3/\epsilon^3)$ as claimed. □

D.5 Extensions

D.5.1 Stochastic Costs

Suppose each cost $c_h(s,a)$ is replaced with a cost distribution $C_h(s,a)$. Here, we consider finitely supported cost distributions whose supports are at most $m \in \mathbb{N}$. Then, instead of the agent occurring cost $c_h(s,a)$ upon taking action a in state s at time h, the agent occurs a random cost $c_h \sim C_h(s,a)$. Generally, this necessitates histories be cost dependent, and so the policy evaluation equations become,

$$V_h^{\pi}(\tau_h) = r_h(s,a) + \sum_{c',s'} C_h(c' \mid s,a) P_h(s' \mid s,a) V_{h+1}^{\pi}(\tau_h, a, c', s'). \tag{CPE}$$

Cover MDP. This implicitly changes the definition of $\mathcal{V}$ since the histories considered in the definition must now include cost history. Since the cost distributions are finitely supported, $\mathcal{V}$ remains a finite set. The main difference for $\bar{M}$ is that the future value demands must depend on both the immediate cost and the next state.

This slightly changes the action space:

$$\bar{\mathcal{A}}_h(s,v) \stackrel{\text{def}}{=} \left\{ (a,\mathbf{v}) \in \mathcal{A} \times \mathcal{V}^{m\times S} \mid r_h(s,a) + \sum_{c',s'} C_h(c' \mid s,a) P_h(s' \mid s,a) v_{c',s'} \geq v \right\}.$$

Bellman Updates. In order to solve $\bar{M}$ using Algorithm 12, we must extend the definition of TSR to also be recursive in the immediate costs. The key difference of the *TSRC* condition is that g's recursion is now two dimensional.

Definition 29 (TSRC)**.** We call a criterion C *time-space-cost-recursive* (TSRC) if $C_M^\pi = C_1^\pi(s_0)$ where $C_{H+1}^\pi(\cdot) = \mathbf{0}$ and for any $h \in [H]$ and $\tau_h \in \mathcal{H}_h$ letting $s = s_h(\tau_h)$ and $a = \pi_h(\tau_h)$,

$$C_h^\pi(\tau_h) = c_h(s,a) + f\left(\left(C_h(c' \mid s,a), P_h(s' \mid s,a), C_{h+1}^\pi(\tau_h, a, c', s') \right)_{c',s'} \right). \quad \text{(D.15)}$$

In the above, $c' \in C_h(s,a)$ and $s' \in P_h(s,a)$. We now require that f be computable in $O(mS)$ time. We also require that the f term above is equal to $g_h^{\tau_h,a}(1,1)$, where, $g_h^{\tau_h,a}(m+1,1) = 0$, $g_h^{\tau_h,a}(k,S+1) = g_h^{\tau_h,a}(k+1,1)$, and,

$$g_h^{\tau_h,a}(k,t) = \alpha\left(\beta\left(C_h(c_k \mid s,a), P_h(t \mid s,a), C_{h+1}^\pi(\tau_h,a,t) \right), g_h^{\tau_h,a}(k,t+1) \right). \quad \text{(D.16)}$$

In the above, we assume c_k is the kth supported cost of $C_h(s,a)$. Again, both α, β can be computed in $O(1)$ time, but now $\alpha(\beta(y,\cdot),x) = x$ whenever $0 \in y$.

Our examples from before also carry over to the stochastic cost setting.

Proposition 19 (TSCR examples)**.** *The following classical constraints can be modeled by a TSCR cost constraint.*

1. (Expectation Constraints) *We capture these constraints by defining* $\alpha(x,y) \stackrel{def}{=} x + y$ *and* $\beta(x,y,z) \stackrel{def}{=} xyz$.

2. (Almost Sure Constraints) *We capture these constraints by defining* $\alpha(x,y) \stackrel{def}{=} \max(x,y)$ *and* $\beta(x,y,z) \stackrel{def}{=} [x>0 \wedge y>0]z$.

3. (Anytime Constraints) *We capture these constraints by defining* $\alpha(x,y) \stackrel{def}{=} \max(0,\max(x,y))$ *and* $\beta(x,y,z) \stackrel{def}{=} [x>0 \wedge y>0]z$.

We can then modify our approximate recursion from before.

Definition 30. We define, $\hat{g}^{s,a}_{h,v}(m{+}1,1,u) \stackrel{\text{def}}{=} \chi_{\{u \geq v\}}$, $\hat{g}^{s,a}_{h,v}(k,S{+}1,u) \stackrel{\text{def}}{=} \hat{g}^{s,a}_{h,v}(k{+}1,1,u)$ and for $t \leq S$,

$$\begin{aligned}\hat{g}^{s,a}_{h,\hat{v}}(k,t,u) \stackrel{\text{def}}{=} \min_{v_{k,t} \in \mathcal{V}} \alpha\Big(&\beta\left(C_h(c_k \mid s,a), P_h(t \mid s,a), \bar{C}^*_{h+1}(t, v_{k,t})\right), \\ &\hat{g}^{s,a}_{h,v}\left(k, t+1, \lfloor u + C_h(c_k \mid s,a) P_h(t \mid s,a) v_{k,t} \rfloor_{\mathcal{G}}\right)\Big).\end{aligned} \tag{D.17}$$

Approximation. Lastly, our rounding now occur error over time, space, and cost. Thus, we simply need to slightly modify our rounding functions. The main change is we use $\delta \stackrel{\text{def}}{=} \frac{\epsilon}{H(mS+1)+1}$. We also further relax our lower bounds to $\kappa(v) \stackrel{\text{def}}{=} v - \delta(mS{+}1)$ and $\kappa \stackrel{\text{def}}{=} v(1-\delta)^{mS+1}$ respectively. Our running times correspondingly will have m^3 terms now.

D.5.2 Infinite Discounting

Approximations. Since we focus on approximation algorithms, the infinite discounted case can be immediately handled by using the idea of effective horizon. We can treat the problem as a finite horizon problem where the finite horizon H defined so that $\sum_{h=H}^{\infty} \gamma^{h-1} r_{max} \leq \epsilon'$. By choosing ϵ' and ϵ small enough, we can get traditional value approximations. The discounting also ensures the effective horizon H is polynomially sized implying efficient computation. We just need to assume that 0-cost actions are always available so that the policy can guarantee feasibility after the effective horizon has passed.

Hardness. We also note that all of the standard hardness results concerning deterministic policy computation carries over to the infinite discounting case even if all quantities are stationary.

D.5.3 Faster Approximations

We can significantly improve the running time of our FPTAS. The main guarantee is given in Corollary 8. They key step is to modify Algorithm 11 to use the differences instead of the sums. It is easy to see that this is equivalent since,

$$r_h(s,a) + \sum_{s'} P_h(s' \mid s,a) v_{s'} \geq v \iff v - \sum_{s'} P_h(s' \mid s,a) v_{s'} \leq r_h(s,a).$$

Since rounding down the differences make them larger, it becomes harder to be below $r_h(s,a)$. Consequently, we now interpret κ as an upper bound for $r_h(s,a)$ instead of a lower bound on v The approximate dynamic programming method based on differences can be seen in Definition 31.

Definition 31. Fix a rounding down function $\lfloor \cdot \rfloor_{\mathcal{G}}$ and upper bound function κ. For any $h \in [H]$, $s \in \mathcal{S}$, $v \in \mathcal{V}$, and $u \in \mathbb{R}$, we define, $\hat{g}^{s,a}_{h,v}(S+1,u) = \chi_{\{u \leq \kappa(r_h(s,a))\}}$ and for $t \leq S$,

$$\hat{g}^{s,a}_h(t,u) \stackrel{\text{def}}{=} \min_{v_t \in \mathcal{V}} \alpha\left(\beta\left(P_h(t \mid s,a), \bar{C}^*_{h+1}(t,v_t)\right), \hat{g}^{s,a}_h(t+1, \lfloor u - P_h(t \mid s,a) v_t \rfloor_{\mathcal{G}}\right). \tag{DIF}$$

The recursion is nearly identical to the originally, and unsurprisingly, it retains the same theoretical guarantees but in the reverse order. The guarantees can be seen in Lemma 31, which is straightforward to prove following the approach in the proof of Lemma 11.

Algorithm 19 Approx Solve

Input: $(\bar{M}, \bar{C})$

$\hat{C}^*_{H+1}(s,v) \leftarrow \chi_{\{v \le 0\}}$ for all $(s,v) \in \bar{\mathcal{S}}$
for $h \leftarrow H$ down to 1 **do**
  for $s \in \mathcal{S}$ **do**
    for $a \in \mathcal{A}$ **do**
      $\hat{g}^{s,a}_h(S+1,u) \leftarrow \chi_{\{u \le \kappa(r_h(s,a))\}}\ \forall u \in \hat{\mathcal{U}}^{s,a}_h$
      for $t \leftarrow S$ down to 1 **do**
        for $u \in \hat{\mathcal{U}}^{s,a}_h$ **do**
          $\hat{v}_{t,a}\hat{g}^{s,a}_h(t,u) \leftarrow$ (DIF)
    for $v \in \mathcal{V}$ **do**
      $a^*, \hat{C}^*_h(s,v) \leftarrow \min_{a \in \mathcal{A}} c_h(s,a) + \hat{g}^{s,a}_h(1,v)$
      $\pi_h(s,v) \leftarrow a^*$
return π and $\hat{C}^*$

Lemma 31. *For any $t \in [S+1]$ and $u \in \mathbb{R}$, we have that,*

$$\hat{g}^{s,a}_h(t,u) = \min_{\mathbf{v} \in \mathcal{V}^{S-t+1}} \hat{g}^{s,a}_{h,\hat{\mathbf{v}}}(t) \quad s.t. \quad \tilde{\sigma}^{s,a}_{h,\mathbf{v}}(t,u) \le \kappa(r_h(s,a)), \tag{D.18}$$

where $\tilde{\sigma}^{s,a}_{h,\mathbf{v}}(t,u) \overset{def}{=} \lfloor \lfloor u - P_h(t \mid s,a)v_t \rfloor_{\mathcal{G}} - \ldots - P_h(S \mid s,a)v_S \rfloor_{\mathcal{G}}$.

The difference version so far does not help us get faster algorithms. The key is in how we use it. Since the base case of the recursion is $r_h(s,a)$ and not v, we can compute the approximate bellman update for all v's simultaneously. This ends up saving us a factor of $|\mathcal{V}|$ that we had in the original Algorithm 12. The new algorithm is defined in Algorithm 19. The inputs to the recursion are define in Definition 32.

Definition 32. For any $h \in [H]$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$, we define $\hat{\mathcal{U}}^{s,a}_h(1) \overset{\text{def}}{=} \mathcal{V}$ and for any $t \in [S]$,

$$\hat{\mathcal{U}}^{s,a}_h(t+1) \overset{\text{def}}{=} \bigcup_{v_t \in \mathcal{V}} \bigcup_{\hat{\sigma} \in \hat{\mathcal{U}}^{s,a}_h(t)} \left\{ \lfloor \hat{\sigma} - P_h(t \mid s,a)v_t \rfloor_{\mathcal{G}} \right\}. \tag{D.19}$$

Proposition 20. *The running time of Algorithm 19 is $O(HS^2A|\mathcal{V}|\hat{\sigma})$.*

Corollary 8 (Running Time Improvements)**.** *Using Algorithm 19, the running time of our additive-FPTAS becomes* $O(H^5S^4Ar_{max}^2/\epsilon^2)$*, and the running time of our relative-FPTAS becomes* $O(H^5S^4A\log(\frac{r_{max}}{r_{min}p_{min}})^2/\epsilon^2)$

Approximation Details. Although the running times our clear from removing the factor of $|\hat{V}|$, we need to slightly alter our approximation schemes for this to work. First, we need to use $\kappa(r_h(s,a)) \stackrel{\text{def}}{=} r_h(s,a) + \delta$ for the additive approximation. The proof from before goes through almost identically.

However, for the relative approximation, no choice of upper bound can ensure enough feasibility. Thus, we simply use $\kappa(r_h(s,a)) \stackrel{\text{def}}{=} r_h(s,a)$ and apply a different analysis. We also note that technically, differences can become negative. To deal with this the relative rounding function should simply send any negative number to 0: $\lfloor -x \rfloor_{\mathcal{G}} \stackrel{\text{def}}{=} 0$. The analysis is mostly the same, but the feasibility statement must be slightly modified.

Lemma 32. *Suppose all rewards are non-negative. For any* $h \in [H+1]$ *and* $(s,v) \in \bar{\mathcal{S}}$, $\hat{C}_h^*(s, \lfloor v(1-\delta)^{H-h+1} \rfloor_{\mathcal{G}}) \leq \bar{C}_h^*(s,v)$.

The idea is that since no fixed upper bound can capture arbitrary input values, we simply input relative values. Then, the feasibility part of Lemma 29 goes through as before. The proofs mostly remain the same, but the rounding must again change. We must now start at the smaller v_{min} that is the original v_{min} scaled by a factor of $(1-\delta)^H$ to ensure that $\lfloor V_M^*(1-\delta)^H \rfloor$ is in $\hat{V}$. This makes $\hat{V}$ larger, but not by too much as we argued in previous analyses.

Appendix E

Chapter 6 Appendix

E.1 Proofs for Section 6.2

E.1.1 Proof of Proposition 17

Proof.

Expectation Constraints. We define $C_M^\pi \stackrel{\text{def}}{=} \mathbb{E}_M\left[\sum_{h=1}^H c_H\right]$. Under this definition, the standard policy evaluation equations imply that,

$$C_h^\pi(\tau_h) = c_h(s,a) + \sum_{s'} P_h(s' \mid s,a) C_{h+1}^\pi(\tau_{h+1}). \tag{E.1}$$

It is then clear that this can be written in (f,g)-form for f being summation and g being the identity. It is easy to see that these functions have the desired properties.

Chance Constraints. Let M^0 denote the initial caMDP. We define $C_{M^0}^\pi \stackrel{\text{def}}{=} \mathbb{P}_M^\pi\left[\sum_{h=1}^H c_h > B\right]$. We see that the probability can be recursively decomposed as

follows for the anytime variant:

$$C_h^\pi(\tau_h, \bar{c}) = [c_h(s,a) + \bar{c} > B] + \sum_{s'} P_h(s' \mid s,a) C_{h+1}^\pi(\tau_{h+1}, c_h(s,a) + \bar{c}). \tag{E.2}$$

For the general invariant, we only include the indicator term at step H. To write this into the desired form, we can define a cost-augmented MDP M that keeps track of the cumulative cost at each step as in [77]. In particular, the anytime variant has the immediate cost defined to be $c_h((s,\bar{c}),a) \stackrel{\text{def}}{=} [c_h(s,a) + \bar{c} > B]$. Then, it is clear that the expected cost for the new M exactly corresponds to the probability cost. Thus, the claim holds.

Almost-sure Constraints. We define $C_M^\pi \stackrel{\text{def}}{=} \max_{\substack{\tau_{H+1} \\ \mathbb{P}_M^\pi[\tau_{H+1}]>0}} \left[\sum_{h=1}^H c_H\right]$ to be the worst case cost. Under this definition, it is known that the worst-case cost decomposes into,

$$C_h^\pi(\tau_h) = c_h(s,a) + \max_{s'}[P_h(s' \mid s,a) > 0] C_{h+1}^\pi(\tau_{h+1}). \tag{E.3}$$

It is then clear that this can be written in (f,g)-form for f being maximum and g being the indicator. Properties of maximum imply that $\max_{s'}(C(s') + \epsilon) \leq \max_{s'} C(s') + \epsilon$. Thus, the total combination is a short map, and the rest of the needed properties can be seen to hold. The anytime variant follows similarly.

□

E.1.2 Proof of Theorem 14

Proof. The theorem follows immediately by translating the results on the SR-criterion into their original forms in the proof above. □

E.2 Proof for Section 6.3

E.2.1 Helpful Technical Lemmas

Definition 33 (Budget Space)**.** For any $s \in \mathcal{S}$, we define $\mathcal{B}_{H+1}(s) \stackrel{\text{def}}{=} \{0\}$, and for any $h \in [H]$,

$$\mathcal{B}_h(s) \stackrel{\text{def}}{=} \bigcup_a \bigcup_{\mathbf{b} \in \times_{s'} \mathcal{B}_{h+1}(s')} \left\{ c_h(s,a) + \underset{s'}{f} g(P_h(s' \mid s,a), b_{s'}) \right\}. \tag{E.4}$$

We define $\mathcal{B} \stackrel{\text{def}}{=} \bigcup_{h,s} \mathcal{B}_h(s)$.

Lemma 33 (Budget Space Intution)**.** *For all $s \in \mathcal{S}$ and $h \in [H+1]$,*

$$\mathcal{B}_h(s) = \left\{ b \in \mathbb{R}^d \mid \exists \pi \in \Pi^D, \tau_h \in \mathcal{H}_h,\ (s = s_h(\tau_h) \wedge C_h^\pi(\tau_h) = b) \right\}, \tag{E.5}$$

and $|\mathcal{B}_h(s)| \leq A^{\sum_{t=h}^H S^{H-t}}$. Thus, $\mathcal{B}$ can be computed in finite time using backward induction.

Proof. We proceed by induction on h. Let $s \in \mathcal{S}$ be arbitrary.

Base Case. For the base case, we consider $h = H+1$. In this case, we know that for any $\pi \in \Pi^D$ and any $\tau \in \mathcal{H}_{H+1}$, $C_{H+1}^\pi(\tau_{H+1}) = 0 \in \{0\} = \mathcal{B}_{H+1}(s)$ by definition. Furthermore, $|\mathcal{B}_{H+1}(s)| = 1 = A^0 = A^{\sum_{t=H+1}^H S^t}$.

Inductive Step. For the inductive step, we consider $h \leq H$. In this case, we know that for any $\pi \in \Pi^D$ and any $\tau_h \in \mathcal{H}_h$, if $s = s_h(\tau_h)$ and $a = \pi_h(\tau_h)$, then the policy evaluation equations imply,

$$C_h^\pi(\tau_h) = c_h(s,a) + \underset{s'}{f} g(P_h(s' \mid s,a), C_{h+1}^\pi(\tau_h, a, s')).$$

We know by the induction hypothesis that $V^{\pi}_{h+1}(\tau_h, a, s') \in \mathcal{B}_{h+1}(s')$. Thus, by (E.4), $C^{\pi}_h(\tau_h) \in \mathcal{B}_h(s)$. Lastly, we see by (E.4) and the induction hypothesis that,

$$|\mathcal{B}_h(s)| \leq A \prod_{s'} |\mathcal{B}_{h+1}(s')| \leq A \prod_{s'} A^{\sum_{t=h+1}^{H} S^{H-t}} = A^{1+S\sum_{t=h+1}^{H} S^{H-t}} = A^{\sum_{t=h}^{H} S^{H-t}}.$$

This completes the proof. □

E.2.2 Proof of Lemma 13

Proof. First, let $V^*_h(\tau_h, b)$ denote the supremum in (6.5). We proceed by induction on h.

Base Case. For the base case, we consider $h = H + 1$. Definition 19 implies that $C^{\pi}_{H+1}(\tau_{H+1}) = 0$ for any $\pi \in \Pi^D$. Thus, there exists a $\pi \in \Pi^D$ satisfying $C^{\pi}_{H+1}(\tau_{H+1}) \leq b$ if and only if $b \geq 0$. We also know by definition that any policy π satisfies $V^{\pi}_{H+1}(\tau_{H+1}) = 0$ and if no feasible policy exists $V^*_{H+1}(\tau_{H+1}, b) = -\infty$ by convention. Therefore, we see that $V^*_{H+1}(\tau_{H+1}, b) = -\chi_{\{b \geq 0\}}$. Then, by definition of $\bar{V}^*_{H+1}$, it follows that,

$$\bar{V}^*_{H+1}(s, b) = -\chi_{\{b \geq 0\}} = V^*_{H+1}(\tau_{H+1}, b).$$

Inductive Step. For the inductive step, we consider any $h \leq H$. If $V^*_h(\tau_h, b) = -\infty$, then trivially $\bar{V}^*_h(s, b) \geq V^*_h(\tau_h, b)$. Instead, suppose that $V^*_h(\tau_h, b) > -\infty$. Then, there must exist a $\pi \in \Pi^D$ satisfying $C^{\pi}_h(\tau_h) \leq b$. Let $a^* = \pi_h(\tau_h)$. By (SR), we know that,

$$C^{\pi}_h(\tau_h) = c_h(s, a^*) + \int_{s'} g\left(P_h(s' \mid s, a^*)\right) C^{\pi}_{h+1}(\tau_h, a^*, s').$$

For each $s' \in \mathcal{S}$, define $b^*_{s'} \stackrel{\text{def}}{=} C^{\pi}_{h+1}(\tau_h, a^*, s')$ and observe that $b^*_{s'} \in \mathcal{B}$. Thus, we see that $(a^*, \mathbf{b}^*) \in \mathcal{A} \times \mathcal{B}^S$ and $c_h(s,a) + f_{s'}\, g(P_h(s' \mid s,a))b_{s'} \leq b$, so $(a^*, \mathbf{b}^*) \in \bar{\mathcal{A}}_h(s,b)$ by definition.

Since π satisfies $C^{\pi}_{h+1}(\tau_h, a^*, s') \leq b^*_{s'}$, we see that $V^*_{h+1}(s', b^*_{s'}) \geq V^{\pi}_{h+1}(\tau_h, a^*, s')$. Thus, the induction hypothesis implies $\bar{V}^*_{h+1}(s', b^*_{s'}) \geq V^*_{h+1}(s', b^*_{s'}) \geq V^{\pi}_{h+1}(\tau_h, a^*, s')$. The optimality equations for $\bar{M}$ then give us,

$$\begin{aligned}
\bar{V}^*_h(s,b) &= \max_{\bar{a} \in \bar{\mathcal{A}}_h(s,b)} \bar{r}_h((s,b),\bar{a}) + \sum_{\bar{s}'} \bar{P}_h(\bar{s}' \mid (s,b),\bar{a})\bar{V}^*_{h+1}(\bar{s}') \\
&= \max_{(a,\mathbf{b}) \in \bar{\mathcal{A}}_h(s,b)} r_h(s,a) + \sum_{s'} P_h(s' \mid s,a)\bar{V}^*_{h+1}(s', b_{s'}) \\
&\geq r_h(s,a^*) + \sum_{s'} P_h(s' \mid s,a^*)\bar{V}^*_{h+1}(s', b^*_{s'}) \\
&\geq r_h(s,a^*) + \sum_{s'} P_h(s' \mid s,a^*)V^{\pi}_{h+1}(\tau_h, a, s') \\
&= V^{\pi}_h(\tau_h).
\end{aligned}$$

The second line used the definition of each quantity in $\bar{M}$. The first inequality used the fact that $(a^*, \mathbf{b}^*) \in \bar{\mathcal{A}}_h(s,b)$. The second inequality used the induction hypothesis. The final equality used the deterministic policy evaluation equations.

Since π was an arbitrary feasible policy for the optimization defining $V^*_h(\tau_h, b)$, we see that $\bar{V}^*_h(s,b) \geq V^*_h(\tau_h, b)$. This completes the proof.

□

E.2.3 Proof of Lemma 14

Proof. We proceed by induction on h.

Base Case. For the base case, we consider $h = H+1$. By definition and assumption, $\bar{V}^{\pi}_{H+1}(s,b) = -\chi_{\{b\geq 0\}} > -\infty$. Thus, it must be the case that $b \geq 0$ and so by Definition 19 $\bar{C}^{\pi}_{H+1}(s,b) = 0 \leq b$.

Inductive Step. For the inductive step, we consider any $h \leq H$. We decompose $\pi_h(s,b) = (a, \mathbf{b})$ where we know $(a, \mathbf{b}) \in \bar{\mathcal{A}}_h(s,b)$ since $\bar{V}^{\pi}_h(s,b) > -\infty$ [1]. Moreover, it must be the case that for any $s' \in \mathcal{S}$ with $P_h(s' \mid s,a) > 0$ that $\bar{V}^{\pi}_{h+1}(s', b_{s'}) > -\infty$ otherwise the average reward would be $-\infty$ which would imply a contradiction:

$$\begin{aligned}
\bar{V}^{\pi}_h(s,b) &= r_h(s,a) + \sum_{s'} P_h(s' \mid s,a) \bar{V}^{\pi}_{h+1}(s', b_{s'}) \\
&= r_h(s,a) + \ldots + P_h(s' \mid s,a)(-\infty) + \ldots \\
&= -\infty.
\end{aligned}$$

Thus, the induction hypothesis implies that $\bar{C}^{\pi}_{h+1}(s', b_{s'}) \leq b_{s'}$ for any such $s' \in \mathcal{S}$. By (SR), we see that,

$$\begin{aligned}
\bar{C}^{\pi}_h(s,b) &= c_h(s,a) + \underset{s'}{f}\, g(P_h(s' \mid s,a)) \bar{C}^{\pi}_{h+1}(s', b_{s'}) \\
&\leq c_h(s,a) + \underset{s'}{f}\, g(P_h(s' \mid s,a)) b_{s'} \\
&\leq b.
\end{aligned}$$

The second line used the fact that f is non-decreasing and g is a non-negative scalar. The third line used the fact that $(a, \mathbf{b}) \in \bar{\mathcal{A}}_h(s,b)$. This completes the proof. □

[1] By convention, we assume $\max \varnothing = -\infty$

E.2.4 Proof of Theorem 15

Proof. If $\bar{V}_1^*(s_0, B) = -\infty$, then we know by Lemma 13 that,

$$-\infty = \bar{V}_1^*(s_0, B) \geq \sup_{\pi \in \Pi^D} V_1^\pi(s_0) \quad \text{s.t. } C_1^\pi(s_0) \leq B. \tag{E.6}$$

In other words, no feasible π exists, so Algorithm 14 reporting "Infeasible" is correct. On the other hand, suppose that $\bar{V}_1^*(s_0, B) > -\infty$ and let π^* be any solution to the optimality equations for $\bar{M}$. By Lemma 14, we know that $C_1^\pi(s_0, B) \leq B$ implying that π^* is a feasible solution. Moreover, Lemma 13 again tells us that,

$$\bar{V}_1^{\pi^*}(s_0, B) = \bar{V}_1^*(s_0, B) \geq \sup_{\pi \in \Pi^D} V_1^\pi(s_0) \quad \text{s.t. } C_1^\pi(s_0) \leq B. \tag{E.7}$$

Thus, π^* is an optimal solution to (CON) and Algorithm 14 correctly returns it. Therefore, in all cases, Algorithm 14 is correct. □

E.3 Proofs for Section 6.4

Formally, $\hat{f}_{h,\mathbf{b}}^{s,a}(t, \hat{F}) \stackrel{\text{def}}{=} \hat{f}_{h,\mathbf{b}}^{s,a}\left(t+1, \lfloor \ell \rfloor_{\mathcal{G}} f(\hat{F}, g(P_h(t \mid s, a))b_t)\right)$ is recursively defined with base case $\hat{f}_{h,\mathbf{b}}^{s,a}(S+1, \hat{F}) \stackrel{\text{def}}{=} \hat{F}$.

E.3.1 Proof of Lemma 15

Proof. First, we show that,

$$\begin{aligned} \bar{V}_{h,b}^{s,a}(t, \hat{F}) = \max_{\mathbf{b} \in \mathcal{B}^{S-t+1}} & \sum_{s'=t}^{S} P_h(s' \mid s, a) \bar{V}_{h+1}^{*}(s', b_{s'}) \\ \text{s.t.} \quad & c_h(s, a) + f_{h,\mathbf{b}}^{s,a}(t, F) \leq b, \end{aligned} \tag{E.8}$$

For notational simplicity, we define $\bar{V}_{h,\mathbf{b}}^{s,a}(t) \stackrel{\text{def}}{=} \sum_{s'=t}^{S} P_h(s' \mid s, a) \bar{V}_{h+1}^{*}(s', b_{s'})$. We proceed by induction on t.

Base Case. For the base case, we consider $t = S + 1$. By definition, we know that $\bar{V}_{h,b}^{s,a}(t, F) = -\chi_{\{c_h(s,a)+F \leq b\}}$. We just need to show that the maximum in (E.8) also matches this expression. First, observe objective is the empty summation, which is 0. Also, $f_{h,\mathbf{b}}^{s,a}(S + 1, F) = F$, so the constraint is satisfied iff $c_h(s, a) + F \leq b$. Thus, the maximum is 0 when $c_h(s, a) + F \leq b$ and is $-\infty$ due to infeasibility otherwise. In other words, it equals $-\chi_{\{c_h(s,a)+F \leq b\}}$ as was to be shown.

Inductive Step. For the inductive step, we consider any $t \leq S$. From (6.6), we see that,

$$\begin{aligned}
\bar{V}^{s,a}_{h,b}(t,F) &= \max_{b_t \in \mathcal{B}} P_h(t \mid s,a)\bar{V}^*_{h+1}(t,b_t) + \bar{V}^{s,a}_{h,b}\left(t+1, f\left(F, g(P_h(t \mid s,a))b_t\right)\right) \\
&= \max_{b_t \in \mathcal{B}} P_h(t \mid s,a)\bar{V}^*_{h+1}(t,b_t) + \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t}, \\ c_h(s,a)+f^{s,a}_{h,\mathbf{b}}(t+1, f(F, g(P_h(t|s,a))b_t)) \leq b}} \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{b_t \in \mathcal{B}} \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t}, \\ c_h(s,a)+f^{s,a}_{h,\mathbf{b}}(t+1, f(F, g(P_h(t|s,a))b_t)) \leq b}} P_h(t \mid s,a)\bar{V}^*_{h+1}(t,b_t) + \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t+1}, \\ c_h(s,a)+f^{s,a}_{h,\mathbf{b}}(t+1, f(F, g(P_h(t|s,a))b_t)) \leq b}} P_h(t \mid s,a)\bar{V}^*_{h+1}(t,b_t) + \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t+1}, \\ c_h(s,a)+f^{s,a}_{h,\mathbf{b}}(t,F) \leq b}} P_h(t \mid s,a)\bar{V}^*_{h+1}(t,b_t) + \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t+1}, \\ c_h(s,a)+f^{s,a}_{h,\mathbf{b}}(t,F) \leq b}} \bar{V}^{s,a}_{h,\mathbf{b}}(t)
\end{aligned}$$

The second line used the induction hypothesis. The third line used the fact that the first term is independent of future b values. The fourth line used the properties of maximum. The fourth line used the recursive definition of $f^{s,a}_{h,\mathbf{b}}(t,F)$. The last line used the recursive definition of $\bar{V}^{s,a}_{h,\mathbf{b}}(t)$.

For the second claim, we observe that,

$$\begin{aligned}
\bar{V}_h^*(s,b) &= \max_{\substack{a,\mathbf{b},\\ c_h(s,a)+f_{s'}\, g(P_h(s'|s,a))b_{s'}\le b}} r_h(s,a) + \sum_{s'} P_h(s' \mid s,a)\bar{V}_{h+1}^*(s',b_{s'}) \\
&= \max_{\substack{a,\mathbf{b},\\ c_h(s,a)+f_{h,\mathbf{b}}^{s,a}(1,0)\le b}} r_h(s,a) + \sum_{s'} P_h(s' \mid s,a)\bar{V}_{h+1}^*(s',b_{s'}) \\
&= \max_a \max_{\substack{\mathbf{b},\\ c_h(s,a)+f_{h,\mathbf{b}}^{s,a}(1,0)\le b}} r_h(s,a) + \sum_{s'} P_h(s' \mid s,a)\bar{V}_{h+1}^*(s',b_{s'}) \\
&= \max_a r_h(s,a) + \max_{\substack{\mathbf{b},\\ c_h(s,a)+f_{h,\mathbf{b}}^{s,a}(1,0)\le b}} \sum_{s'} P_h(s' \mid s,a)\bar{V}_{h+1}^*(s',b_{s'}) \\
&= \max_a r_h(s,a) + \bar{V}_{h,b}^{s,a}(1,0).
\end{aligned}$$

□

E.3.2 Proof of Lemma 16

Proof. Recall, as in the proof of Lemma 15, we define $\bar{V}_{h,\mathbf{b}}^{s,a}(t) \stackrel{\text{def}}{=} \sum_{s'=t}^{S} P_h(s' \mid s,a)\bar{V}_{h+1}^*(s',b_{s'})$ to simplify expressions. We proceed by induction on t.

Base Case. For the base case, we consider $t = S+1$. By definition, we know that $\hat{V}_{h,b}^{s,a}(t,\hat{F}) = -\chi_{\{c_h(s,a)+\hat{F}\le\kappa(b)\}}$. We just need to show that the maximum in (6.7) also matches this expression. First, observe objective is the empty summation, which is 0. Also, $\hat{f}_{h,\mathbf{b}}^{s,a}(S+1,F) = F$, so the constraint is satisfied iff $c_h(s,a)+\hat{F} \le \kappa(b)$. Thus, the maximum is 0 when $c_h(s,a)+\hat{F} \le \kappa(b)$ and is $-\infty$ due to infeasibility otherwise. In other words, it equals $-\chi_{\{c_h(s,a)+\hat{F}\le\kappa(b)\}}$ as was to be shown.

Inductive Step. For the inductive step, we consider any $t \leq S$. From (ADP), we see that, $\hat{V}^{s,a}_{h,b}(t, \hat{F}) =$

$$
\begin{aligned}
&\max_{b_t \in \mathcal{B}} P_h(t \mid s, a) \bar{V}^*_{h+1}(t, b_t) + \hat{V}^{s,a}_{h,b}\left(t+1, \left\lceil f\left(\hat{F}, g(P_h(t \mid s, a))b_t\right)\right\rceil_\ell\right) \\
&= \max_{b_t \in \mathcal{B}} P_h(t \mid s, a) \bar{V}^*_{h+1}(t, b_t) + \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t}, \\ c_h(s,a) + \hat{f}^{s,a}_{h,\mathbf{b}}\left(t+1, \left\lceil f\left(\hat{F}, g(P_h(t|s,a))b_t\right)\right\rceil_\ell\right) \leq \kappa(b)}} \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{b_t \in \mathcal{B}} \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t}, \\ c_h(s,a) + \hat{f}^{s,a}_{h,\mathbf{b}}\left(t+1, \left\lceil f\left(\hat{F}, g(P_h(t|s,a))b_t\right)\right\rceil_\ell\right) \leq \kappa(b)}} P_h(t \mid s, a) \bar{V}^*_{h+1}(t, b_t) + \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t+1}, \\ c_h(s,a) + \hat{f}^{s,a}_{h,\mathbf{b}}\left(t+1, \left\lceil f\left(\hat{F}, g(P_h(t|s,a))b_t\right)\right\rceil_\ell\right) \leq \kappa(b)}} P_h(t \mid s, a) \bar{V}^*_{h+1}(t, b_t) + \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t+1}, \\ c_h(s,a) + \hat{f}^{s,a}_{h,\mathbf{b}}\left(t, \hat{F}\right) \leq \kappa(b)}} P_h(t \mid s, a) \bar{V}^*_{h+1}(t, b_t) + \bar{V}^{s,a}_{h,\mathbf{b}}(t+1) \\
&= \max_{\substack{\mathbf{b} \in \mathcal{B}^{S-t+1}, \\ c_h(s,a) + \hat{f}^{s,a}_{h,\mathbf{b}}\left(t, \hat{F}\right) \leq \kappa(b)}} \bar{V}^{s,a}_{h,\mathbf{b}}(t)
\end{aligned}
$$

The second line used the induction hypothesis. The third line used the fact that the first term is independent of future b values. The fourth line used the properties of maximum. The fourth line used the recursive definition of $\hat{f}^{s,a}_{h,\mathbf{b}}(t, \hat{F})$. The last line used the recursive definition of $\bar{V}^{s,a}_{h,\mathbf{b}}(t)$.

For the second claim, we simply observe without rounding that (ADP) is the same as (6.6). Thus, Lemma 15 yields the result. □

E.3.3 Proof of Theorem 16

Proof. The fact that Algorithm 16 correctly solves any $\bar{M}$ follows from the fact that (AU) is equivalent to (BU) via Lemma 16.

For the time complexity claim, observe that the number of subproblems considered is $O(HS^2A|\mathcal{B}||\hat{\mathcal{F}}|)$ and the time needed per subproblem is $O(|\mathcal{B}|)$ to explicitly

optimize each artificial budget. Thus, the running time is $O(HS^2A|\mathcal{B}|^2|\hat{\mathcal{F}}|)$. We can further analyze $|\hat{\mathcal{F}}|$ in terms of the original input variables. First, we claim that $\hat{\mathcal{F}} \subseteq [b_{min}, b_{max} + \ell S]$. To see this, observe that the rounded input at state $t+1$ is,

$$f(\hat{F}, \lceil b_t \rceil_\ell) \geq f(F, b_t) = \sum_{s'=1}^{t} g(P_h(s' \mid s, a))b_{s'} \geq \sum_{s'=1}^{t} g(P_h(s' \mid s, a))b_{min} \geq b_{min}.$$

Here, we used the fact that f is non-decreasing and the weighted combination is a short map rooted at 0. Similarly, we see,

$$\begin{aligned} f(\hat{F}, \lceil b_t \rceil_\ell) &\leq f(F, \lceil b_t \rceil_\ell) + \ell(t-1) \\ &\leq \sum_{s'=1}^{t} g(P_h(s' \mid s, a))(b_{s'} + \ell) + \ell(t-1) \\ &\leq \sum_{s'=1}^{t} g(P_h(s' \mid s, a))b_{max} + \ell t \\ &\leq b_{max} + \ell t. \end{aligned}$$

Under this assumption, it is clear that the number of integer multiples of ℓ residing in this superset is $O((b_{max} + \ell S - b_{min})/\ell)$ per constraint. When considering all constraints at once, this becomes $O(\|b_{max} + \ell S - b_{min}\|_\infty^m / \ell^m) = O(\|b_{max} - b_{min}\|_\infty^m / \ell^m + S^m)$. Incorporating this into the runtime gives $O(HS^{m+2}A|\mathcal{B}|^2 \|b_{max} - b_{min}\|_\infty^m / \ell^m)$.

Similar to the reasoning above, we can see the cost of any policy, and thus the artificial budget set, is contained within $[Hc_{min}, Hc_{max}]$. Using this fact, we get the final running time $O(H^{m+1}S^{m+2}A|\mathcal{B}|^2 \|c_{max} - c_{min}\|_\infty^m / \ell^m)$.

□

E.4 Proofs for Section 6.5

E.4.1 Time-Space Error Lemmas

Lemma 34 (Time Error). *For any $h \in [H]$, $a \in \mathcal{A}$, if $\mathbf{b}' \leq \mathbf{b} + x$, then,*

$$f^{s,a}_{h,\mathbf{b}}(1,0) \leq f^{s,a}_{h,\mathbf{b}'}(1,0) \leq f^{s,a}_{h,\mathbf{b}}(1,0) + x. \tag{E.9}$$

Here, we translate a scalar $x > 0$ into the vector $(x, \ldots, x)$.

Proof. By definition of $f^{s,a}_{h,\mathbf{b}'}$,

$$\begin{aligned}
f^{s,a}_{h,\mathbf{b}'}(1,0) &= f(0, \underset{s'}{f}\, g(P_h(s' \mid s,a))b'_{s'}) \\
&= \underset{s'}{f}\, g(P_h(s' \mid s,a))b'_{s'} \\
&\geq \underset{s'}{f}\, g(P_h(s' \mid s,a))b_{s'} \\
&= f(0, \underset{s'}{f}\, g(P_h(s' \mid s,a))b_{s'}) \\
&= f^{s,a}_{h,\mathbf{b}}(1,0).
\end{aligned}$$

The second and fourth lines used the fact that f is identity preserving. The inequality uses the fact that f is non-decreasing and g is a non-negative scalar, so the total weighted combination is also non-decreasing.

Similarly, we see that,

$$\begin{aligned}
f^{s,a}_{h,\mathbf{b}'}(1,0) &= f(0, \underset{s'}{f}\, g(P_h(s' \mid s,a))b'_{s'}) \\
&= \underset{s'}{f}\, g(P_h(s' \mid s,a))b'_{s'} \\
&\leq \underset{s'}{f}\, g(P_h(s' \mid s,a))(b_{s'} + x) \\
&\leq \underset{s'}{f}\, g(P_h(s' \mid s,a))b_{s'} + x \\
&= f(0, \underset{s'}{f}\, g(P_h(s' \mid s,a))b_{s'}) + x \\
&= f^{s,a}_{h,\mathbf{b}}(1,0) + x.
\end{aligned}$$

The second and fifth lines used the fact that f is identity preserving. The first inequality again uses the fact that the weighted combination is non-decreasing. The second inequality follows since the weighted combination is a short map with respect to the infinity norm.

In particular, since $|\alpha(y) - \alpha(z)| \leq \|y - z\|_\infty$ holds for any infinity-norm short map α, we see that $|\alpha(y+z) - \alpha(y)| \leq \|z\|_\infty$. Moreover, if α is non-decreasing and z is a positive scaler treated as a vector, we further have $\alpha(y+z) - \alpha(y) = |\alpha(y+z) - \alpha(y)| \leq \|z\|_\infty = z$. This final inequality immediately implies that $\alpha(y+z) \leq \alpha(y) + z$. When α is vector-valued, this inequality holds component-wise. □

Since f is associative, we can define $f^{s,a}_{h,\mathbf{b}}(t,F) = f(F, f^{S}_{s'=t}\, g(P_h(s' \mid s,a))b_{s'})$ either forward recursively or backward recursively.

Lemma 35 (Space Error)**.** *For any $h \in [H]$, $a \in \mathcal{A}$, $\mathbf{b} \in \mathbb{R}^{m \times S}$, $u \in \mathbb{R}^m$, and $t \in [S+1]$,*

$$f^{s,a}_{h,\mathbf{b}}(t,u) \leq \hat{f}^{s,a}_{h,\mathbf{b}}(t,u) \leq f^{s,a}_{h,\mathbf{b}}(t,u) + (S - t + 1)\ell. \tag{E.10}$$

Proof. We proceed by induction on t.

Base Case. For the base case, we consider $t = S + 1$. By definition, we have that $\hat{f}^{s,a}_{h,\mathbf{b}}(S+1, u) = u = f^{s,a}_{h,\mathbf{b}}(S+1, u)$. Thus, the claim holds.

Inductive Step. For the inductive step, we consider any $t \leq S$. The recursive definition of $\hat{f}^{s,a}_{h,\mathbf{b}}$ implies,

$$\begin{aligned}
\hat{f}^{s,a}_{h,\mathbf{b}}(t, u) &= \hat{f}^{s,a}_{h,\mathbf{b}}(t+1, \lceil f(u, g(P_h(t \mid s, a))b_t)\rceil_\ell) \\
&\geq f^{s,a}_{h,\mathbf{b}}(t+1, \lceil f(u, g(P_h(t \mid s, a))b_t)\rceil_\ell) \\
&= f(\lceil f(u, g(P_h(t \mid s, a))b_t)\rceil_\ell, \overset{S}{\underset{s'=t+1}{f}} g(P_h(s' \mid s, a)b_{s'})) \\
&\geq f(f(u, g(P_h(t \mid s, a))b_t), \overset{S}{\underset{s'=t+1}{f}} g(P_h(s' \mid s, a)b_{s'})) \\
&= f(u, f(g(P_h(t \mid s, a))b_t, \overset{S}{\underset{s'=t+1}{f}} g(P_h(s' \mid s, a)b_{s'}))) \\
&= f^{s,a}_{h,\mathbf{b}}(t, u).
\end{aligned}$$

The first inequality used the induction hypothesis to replace $\hat{f}$ with f, and the second inequality used that f is non-decreasing in either input and $\lceil b_t \rceil_\ell \geq b_t$. The other lines use f's associativity.

Similarly, we see that,

$$\begin{aligned}
\hat{f}^{s,a}_{h,\mathbf{b}}(t,u) &= \hat{f}^{s,a}_{h,\mathbf{b}}(t+1, \lceil f(u, g(P_h(t \mid s,a))b_t)\rceil_\ell) \\
&\le f^{s,a}_{h,\mathbf{b}}(t+1, \lceil f(u, g(P_h(t \mid s,a))b_t)\rceil_\ell) + (S-t)\ell \\
&= f(\lceil f(u, g(P_h(t \mid s,a))b_t)\rceil_\ell, \operatorname*{f}_{s'=t+1}^{S} g(P_h(s' \mid s,a)b_{s'})) + (S-t)\ell \\
&\le f(f(u, g(P_h(t \mid s,a))b_t) + \ell, \operatorname*{f}_{s'=t+1}^{S} g(P_h(s' \mid s,a)b_{s'})) + (S-t)\ell \\
&\le f(f(u, g(P_h(t \mid s,a))b_t), \operatorname*{f}_{s'=t+1}^{S} g(P_h(s' \mid s,a)b_{s'})) + (S-t+1)\ell \\
&= f(u, f(g(P_h(t \mid s,a))b_t, \operatorname*{f}_{s'=t+1}^{S} g(P_h(s' \mid s,a)b_{s'}))) + (S-t+1)\ell \\
&= f^{s,a}_{h,\mathbf{b}}(t,u) + (S-t+1)\ell.
\end{aligned}$$

The first inequality used the induction hypothesis to replace $\hat{f}$ with f. The second inequality used that f is non-decreasing in either input and $\lceil x\rceil_\ell \le x + \ell$. The third inequality used that f is a short map in the first input. The other lines use f's associativity.

This completes the proof. □

E.4.2 Proof of Lemma 17

Proof. We proceed by induction on h.

Base Case. For the base case, we consider $h = H+1$. Since $\lceil b\rceil_\ell \ge b$, we immediately see,

$$\hat{V}^*_{H+1}(s, \lceil b\rceil_\ell) = -\chi_{\{\lceil b\rceil_\ell \ge 0\}} \ge -\chi_{\{b\ge 0\}} = \bar{V}^*_{H+1}(s,b). \tag{E.11}$$

Inductive Step. For the inductive step, we consider any $h \leq H$. If $\bar{V}_h^*(s,b) = -\infty$, then trivially $\hat{V}_h^*(s, \lceil b \rceil_\ell) \geq \bar{V}_h^*(s,b)$. Now, suppose that $\bar{V}_h^*(s,b) > -\infty$. Let π be a solution to the optimality equations for $\bar{M}$. Consequently, we know that $\bar{V}_h^\pi(s,b) = \bar{V}_h^*(s,b) > -\infty$, which implies $(a^*, \mathbf{b}^*) = \pi_h(s,b) \in \bar{\mathcal{A}}_h(s,b)$. By definition of $\bar{\mathcal{A}}_h(s,b)$,

$$c_h(s,a^*) + f_{h,\mathbf{b}^*}^{s,a^*}(1,0) = c_h(s,a^*) + \int_{s'} g(P_h(s' \mid s,a^*)) b_{s'}^* \leq b \leq \lceil b \rceil_\ell . \tag{E.12}$$

For each $s' \in \mathcal{S}$, define $\hat{b}_{s'}^* \stackrel{\text{def}}{=} \lceil b_{s'}^* \rceil_\ell$. We show $(a^*, \hat{\mathbf{b}}_{s'}^*) \in \hat{\mathcal{A}}_h(s, \lceil b \rceil_\ell)$ as follows:

$$\begin{aligned} c_h(s,a^*) + \hat{f}_{h,\hat{\mathbf{b}}^*}^{s,a}(1,0) &\leq c_h(s,a^*) + f_{h,\hat{\mathbf{b}}^*}^{s,a}(1,0) + \ell S \\ &\leq c_h(s,a^*) + f_{h,\mathbf{b}^*}^{s,a}(1,0) + \ell(S+1) \\ &\leq \lceil b \rceil_\ell + \ell(S+1) \\ &= \kappa(\lceil b \rceil_\ell). \end{aligned}$$

The first inequality follows from Lemma 35. The second inequality follows from Lemma 34 with $\hat{\mathbf{b}}^* \leq \mathbf{b}^* + \ell$. The third inequality follows from (E.12). The equality follows by definition of κ. Thus, $(a^*, \hat{\mathbf{b}}_{s'}^*) \in \hat{\mathcal{A}}_h(s, \lceil b \rceil_\ell)$.

Since $b_{s'}^* \in \mathcal{B}$ by definition, the induction hypothesis implies that $\hat{V}_{h+1}^*(s', \hat{b}_{s'}^*) \geq$

$\bar{V}^*_{h+1}(s', b^*_{s'}) = \bar{V}^\pi_{h+1}(s', b^*_{s'})$. The optimality equations for $\hat{M}$ then imply that,

$$\begin{aligned}
\hat{V}^*_h(s, \lceil b \rceil_\ell) &= \max_{(a,\hat{\mathbf{b}}) \in \hat{\mathcal{A}}_h(s,b)} r_h(s,a) + \sum_{s'} P_h(s' \mid s, a) \hat{V}^*_{h+1}\left(s', \hat{b}_{s'}\right) \\
&\geq r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a) \hat{V}^*_{h+1}\left(s', \hat{b}^*_{s'}\right) \\
&\geq r_h(s, a^*) + \sum_{s'} P_h(s' \mid s, a) \bar{V}^\pi_{h+1}\left(s', b^*_{s'}\right) \\
&= \bar{V}^\pi_h(s, b) \\
&= \bar{V}^*_h(s, b).
\end{aligned}$$

The first inequality used the fact that $(a^*, \hat{\mathbf{b}}^*) \in \hat{\mathcal{A}}_h(s, b)$. The second inequality follows from the induction hypothesis. The last two equalities follow from the standard policy evaluation equations and the definition of π, respectively. This completes the proof.

□

E.4.3 Proof of Lemma 18

Proof. We proceed by induction on h.

Base Case. For the base case, we consider $h = H+1$. By definition and assumption, $\hat{V}^\pi_{H+1}(s, \hat{b}) = -\chi_{\{\hat{b} \geq 0\}} > -\infty$. Thus, it must be the case that $\hat{b} \geq 0$ and so by definition $\hat{C}^\pi_{H+1}(s, \hat{b}) = 0 \leq \hat{b}$.

Inductive Step. For the inductive step, we consider any $h \leq H$. As in the proof of Lemma 14, we know that $(a, \hat{\mathbf{b}}) = \pi_h(s, b) \in \hat{\mathcal{A}}_h(s, \hat{b})$ and for any $s' \in \mathcal{S}$ with $P_h(s' \mid s, a) > 0$ that $\hat{V}^\pi_{h+1}(s', b_{s'}) > -\infty$. Thus, the induction hypothesis implies that $\hat{C}^\pi_{h+1}(s', \hat{b}_{s'}) \leq \hat{b}_{s'} + \ell(S+1)(H-h)$ for any such s'. For any other s', we have

$g(P_h(s' \mid s, a)) = g(0) = 0$ by assumption.

Thus, the weighted combination of $\hat{C}^{\pi}_{h+1}(s', \hat{b}_{s'})$ is equal to the weighted combination of $\hat{\mathbf{b}}'$ where $\hat{b}'_{s'} \stackrel{\text{def}}{=} \hat{C}^{\pi}_{h+1}(s', \hat{b}_{s'})$ if $P_h(s' \mid s, a) > 0$ and $\hat{b}'_{s'} \stackrel{\text{def}}{=} 0$ otherwise. Moreover, we have $\hat{\mathbf{b}}' \leq \hat{\mathbf{b}} + \ell(S+1)(H-h)$ since $\ell > 0$. Thus, by (SR),

$$\begin{aligned}
\hat{C}^{\pi}_h(s, \hat{b}) &= c_h(s, a) + \sum_{s'} g(P_h(s' \mid s, a)) \hat{C}^{\pi}_{h+1}(s', \hat{b}_{s'}) \\
&= c_h(s, a) + f^{s,a}_{h,\hat{\mathbf{b}}'}(1, 0) \\
&\leq c_h(s, a) + f^{s,a}_{h,\hat{\mathbf{b}}}(1, 0) + \ell(S+1)(H-h) \\
&\leq \kappa(\hat{b}) + \ell(S+1)(H-h) \\
&= \hat{b} + \ell(S+1)(H-h+1).
\end{aligned}$$

The first inequality used Lemma 34. The second inequality used the fact that $(a, \hat{\mathbf{b}}) \in \mathcal{A}_h(s, \hat{b})$. The last line used the definition of κ. This completes the proof. □

E.4.4 Proof of Theorem 17

Proof. If (CON) is feasible, then inductively we see that $\hat{V}^*_1(s_0, \lceil B \rceil_\ell) > -\infty$. The contrapositive then implies if $\hat{V}^*_1(s_0, \lceil \ell \rceil_2 B) = -\infty$, then (CON) is infeasible. Thus, when Algorithm 17 outputs "Infeasible", it is correct.

On the other hand, suppose $\hat{V}^*_1(s_0, \lceil B \rceil_\ell) > -\infty$ and that π is an optimal solution to $\hat{M}$. By Lemma 17 and Lemma 13, we know that $\hat{V}^{\pi}_1(s_0, \lceil B \rceil_\ell) \geq \bar{V}^{\pi}_1(s_0, B) \geq V^*$. Also, by Lemma 18, we know that $\hat{C}^{\pi}_1(s_0, \lceil B \rceil_\ell) \leq \lceil B \rceil_\ell + \ell(S+1)H \leq B + \ell(1 + (S+1)H)$. Our choice of $\ell = \frac{\epsilon}{1+(S+1)H}$ then implies that $\hat{C}^{\pi} = \hat{C}^{\pi}_1(s_0, \lceil B \rceil_\ell) \leq B + \epsilon$. Thus, π is an $(0, \epsilon)$-additive bicriteria approximation for (CON).

Both cases together imply that Algorithm 17 is a valid $(0, \epsilon)$-bicriteria.

Time Complexity. We see immediately from Theorem 16 that the running time of Algorithm 17 is at most $O\left(H^{2m+1}S^{2m+2}A|\hat{\mathcal{B}}|^2 \left\|c_{max} - c_{min}\right\|_\infty^m / \epsilon^m\right)$. To complete the analysis, we need to bound $|\hat{\mathcal{B}}|$. First, we note $|\hat{\mathcal{B}}|$ is at most the number of integer multiples of ℓ in the range $[b_{min}, b_{max}] \subseteq [Hc_{min}, Hc_{max}]^m$. For any individual constraint, this number is at most $O(H(c_{max} - Hc_{min})/\ell) \leq O(H^2S(c_{max} - c_{min})/\epsilon)$ using the definition of $\ell = \frac{\epsilon}{1+(S+1)H}$. Thus, the total number of rounded artificial budgets is at most $O((H^2S \left\|c_{max} - c_{min}\right\| / \epsilon)^m)$. Squaring this quantity and plugging it back into our original formula yields: $O\left(H^{6m+1}S^{4m+2}A \left\|c_{max} - c_{min}\right\|_\infty^{3m} / \epsilon^{3m}\right)$.

□

E.4.5 Proof of Proposition 18

Proof. We consider a reduction from the Hamiltonian Path problem. The transitions reflect the graph structure, and the actions determine the edge to follow next. To determine if a Hamiltonian path exists, we can simply make an indicator constraint for each node that signals that node has been reached. It is then clear that relaxing the budget constraint does not help since we can always shrink the budget for any given ϵ-slackness. Thus, the claim holds. □

E.4.6 Proof of Lemma 19

Proof. We proceed by induction on h.

Base Case. For the base case, we consider $h = H + 1$. By definition, we have $\tilde{V}^\pi_{H+1}(\tilde{\tau}_{H+1}) = 0 = V^\pi_{H+1}(\tau_{H+1})$ and $\tilde{C}^\pi_{H+1}(\tilde{\tau}_{H+1}) = 0 = C^\pi_{H+1}(\tau_{H+1})$.

Inductive Step. For the inductive step, we consider any $h \leq H$. For simplicity, let $x \stackrel{\text{def}}{=} \ell(\lambda_r + \lambda_p)Hr_{max}(s_{max} - s_{min})$. The standard policy evaluation equations

imply that,

$$
\begin{aligned}
\tilde{V}_h^{\pi}(\tilde{\tau}_h) &= r_h(\lceil s \rceil_\ell, a) + \sum_{\tilde{s}'} \tilde{P}_h(\tilde{s}' \mid \lceil s \rceil_\ell, a) \tilde{V}_{h+1}^{\pi}(\tilde{\tau}_{h+1}) \\
&= r_h(\lceil s \rceil_\ell, a) + \sum_{\tilde{s}'} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \lceil s \rceil_\ell, a) ds' \tilde{V}_{h+1}^{\pi}(\tilde{\tau}_{h+1}) \\
&= r_h(\lceil s \rceil_\ell, a) + \int_{s'} P_h(s' \mid \lceil s \rceil_\ell, a) \tilde{V}_{h+1}^{\pi}(\tilde{\tau}_{h+1}) ds' \\
&\geq r_h(\lceil s \rceil_\ell, a) + \int_{s'} P_h(s' \mid \lceil s \rceil_\ell, a) (V_{h+1}^{\pi}(\tau_{h+1}) - x(H-h)) ds' \\
&= r_h(\lceil s \rceil_\ell, a) + \int_{s'} P_h(s' \mid \lceil s \rceil_\ell, a) V_{h+1}^{\pi}(\tau_{h+1}) ds' - x(H-h) \\
&\geq r_h(s, a) - \ell \lambda_r + \int_{s'} (P_h(s' \mid s, a) - \ell \lambda_p) V_{h+1}^{\pi}(\tau_{h+1}) ds' - x(H-h) \\
&= V_h^{\pi}(\tau_h) - \ell \lambda_r - \ell \lambda_p \int_{s'} V_{h+1}^{\pi}(\tau_{h+1}) ds' - x(H-h) \\
&\geq V_h^{\pi}(\tau_h) - \ell \lambda_r - \ell \lambda_p H r_{max}(s_{max} - s_{min}) - x(H-h) \\
&\geq V_h^{\pi}(\tau_h) - \ell(\lambda_r + \lambda_p) H r_{max}(s_{max} - s_{min}) - x(H-h) \\
&= V_h^{\pi}(\tau_h) - x(H-h+1).
\end{aligned}
$$

If we let $y \stackrel{\text{def}}{=} \ell(\lambda_c + \lambda_p) H c_{max}(s_{max} - s_{min})$, we also see that,

$$
\begin{aligned}
\tilde{C}_h^\pi(\tilde{\tau}_h) &= c_h(\lceil s \rceil_\ell, a) + \tilde{f}_{\tilde{s}'} \tilde{P}_h(\tilde{s}' \mid \lceil s \rceil_\ell, a) \tilde{C}_{h+1}^\pi(\tilde{\tau}_{h+1}) \\
&= c_h(\lceil s \rceil_\ell, a) + \tilde{f}_{\tilde{s}'} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \lceil s \rceil_\ell, a) ds' \tilde{C}_{h+1}^\pi(\tilde{\tau}_{h+1}) \\
&= c_h(\lceil s \rceil_\ell, a) + \int_{s'} P_h(s' \mid \lceil s \rceil_\ell, a) \tilde{C}_{h+1}^\pi(\tilde{\tau}_{h+1}) \\
&\leq c_h(\lceil s \rceil_\ell, a) + \int_{s'} P_h(s' \mid \lceil s \rceil_\ell, a) (C_{h+1}^\pi(\tau_{h+1}) + y(H-h)) \\
&\leq c_h(\lceil s \rceil_\ell, a) + \int_{s'} P_h(s' \mid \lceil s \rceil_\ell, a) C_{h+1}^\pi(\tau_{h+1}) + y(H-h) \\
&\leq c_h(s, a) + \ell\lambda_c + \int_{s'} (P_h(s' \mid s, a) + \ell\lambda_p) C_{h+1}^\pi(\tau_{h+1}) + y(H-h) \\
&= c_h(s, a) + \int_{s'} P_h(s' \mid s, a) C_{h+1}^\pi(\tau_{h+1}) + \ell\lambda_c + \ell\lambda_p \int_{s'} C_{h+1}^\pi(\tau_{h+1}) + y(H-h) \\
&\leq C_h^\pi(\tau_h) + \ell\lambda_c + \ell\lambda_p \int_{s'} H c_{max} + y(H-h) \\
&\leq C_h^\pi(\tau_h) + \ell\lambda_c + \ell\lambda_p (s_{max} - s_{min}) H c_{max} + y(H-h) \\
&\leq C_h^\pi(\tau_h) + \ell(\lambda_c + \lambda_p)(s_{max} - s_{min}) H c_{max} + y(H-h) \\
&= C_h^\pi(\tau_h) + y(H-h+1).
\end{aligned}
$$

We note the above also holds if P is replaced with a $g(P)$ for a sublinear short map g.

For almost-sure constraints, the proof is slightly different since we need to keep the inner integral by definition of the worst-case cost for continuous state spaces.

Letting $y \stackrel{\text{def}}{=} \ell(\lambda_c + \lambda_p) H c_{max}(s_{max} - s_{min})/\tilde{p}_{min}$, the bound then becomes,

$$
\begin{aligned}
\tilde{C}^{\pi}_h(\tilde{\tau}_h) &= c_h(\lceil s \rceil_\ell, a) + \max_{\tilde{s}'}[\tilde{P}_h(\tilde{s}' \mid \lceil s \rceil_\ell, a) > 0]\tilde{C}^{\pi}_{h+1}(\tilde{\tau}_{h+1}) \\
&= c_h(\lceil s \rceil_\ell, a) + \max_{\tilde{s}'}[\int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \lceil s \rceil_\ell, a) ds' > 0]\tilde{C}^{\pi}_{h+1}(\tilde{\tau}_{h+1}) \\
&= c_h(\lceil s \rceil_\ell, a) + \max_{\tilde{s}'} \frac{\int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \lceil s \rceil_\ell, a) ds'}{p_{\tilde{s}'}} \tilde{C}^{\pi}_{h+1}(\tilde{\tau}_{h+1}) \\
&= c_h(\lceil s \rceil_\ell, a) + \max_{\tilde{s}'} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \lceil s \rceil_\ell, a) \tilde{C}^{\pi}_{h+1}(\tilde{\tau}_{h+1}) ds'/p_{\tilde{s}'} \\
&\leq c_h(\lceil s \rceil_\ell, a) + \max_{\tilde{s}'} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \lceil s \rceil_\ell, a)(C^{\pi}_{h+1}(\tau_{h+1}) + y(H-h)) ds'/p_{\tilde{s}'} \\
&\leq c_h(\lceil s \rceil_\ell, a) + \max_{\tilde{s}'} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid \lceil s \rceil_\ell, a) C^{\pi}_{h+1}(\tau_{h+1}) ds'/p_{\tilde{s}'} + y(H-h) \\
&\leq c_h(s, a) + \ell\lambda_c + \max_{\tilde{s}'} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} P_h(s' \mid s, a) C^{\pi}_{h+1}(\tau_{h+1}) ds'/p_{\tilde{s}'} \\
&+ \ell\lambda_p \max_{\tilde{s}'} \int_{s'=\tilde{s}'}^{\tilde{s}'+\ell} C^{\pi}_{h+1}(\tau_{h+1}) ds'/p\tilde{s}' + y(H-h) \\
&\leq c_h(s, a) + \max_{\mathcal{S}' \subseteq \mathcal{S}} \int_{\mathcal{S}'} \frac{P_h(s' \mid s, a)}{p_{\mathcal{S}'}} C^{\pi}_{h+1}(\tau_{h+1}) ds' + \ell\lambda_c + \ell^2\lambda_p H c_{max}/\tilde{p}_{min} \\
&+ y(H-h) \\
&= C^{\pi}_h(\tau_h) + \ell\lambda_c + \ell^2\lambda_p H c_{max}/\tilde{p}_{min} + y(H-h) \\
&\leq C^{\pi}_h(\tau_h) + \ell(\lambda_c + \lambda_p)(s_{max} - s_{min}) H c_{max}/\tilde{p}_{min} + y(H-h) \\
&= C^{\pi}_h(\tau_h) + y(H-h+1).
\end{aligned}
$$

□

E.4.7 Proof of Theorem 18

Proof. The theorem follows immediately from Theorem 17 and Lemma 19. □

E.5 Extensions

Markov Games. It is easy to see that our augmented approach works to compute constrained equilibria. For efficient algorithms, using $-\infty$ to indicate infeasibility becomes problematic. However, we can still use per-stage LP solutions and add a constraint that the equilibrium value must be larger than some very small constant to rule out invalid $-\infty$ solutions. Alternatively, the AND/OR tree approach used in [77] can be applied here to directly compute all the near-feasible states.

Infinite Discounting. Since we focus on approximation algorithms, the infinite discounted case can be immediately handled by using the idea of effective horizon. We can treat the problem as a finite horizon problem where the finite horizon H is defined so that $\sum_{h=H}^{\infty} \gamma^{h-1} c_{max} \leq \epsilon'$. By choosing ϵ' and ϵ small enough, we can get equivalent feasibility approximations. The discounting also ensures the effective horizon H is polynomially sized, implying efficient computation.

Stochastic Policies. For stochastic policies, our approximate results follow from simply replacing each $\max_a$ and $\max_{b_t}$ with a general linear program over a finite distribution, which can be solved in polynomial time.

Stochastic Costs. For finitely-supported cost distributions, all results remain the same except for almost-sure/anytime constraints, which now must be written in the form:

$$C_h^{\pi}(\tau_h) = \max_{c \in \text{Supp}(C_h(s,a))} c + \max_{s'} [P_h(s' \mid s, a) > 0] C_h^{\pi}(\tau_h, a, c, s'). \tag{E.13}$$

Also, note that histories must now be cost-dependent.

Now, we have that future budgets depend on both the next state and the realized

cost, so our (ADP) must now be dependent on both states and immediate costs for subproblems. The construction is similar to the approach in [76].

Bibliography

[1] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 8 2017. URL https://proceedings.mlr.press/v70/achiam17a.html.

[2] M. Aghassi and D. Bertsimas. Robust game theory. *Mathematical Programming*, 107(1):231–273, 2006. doi: 10.1007/s10107-005-0686-0. URL https://doi.org/10.1007/s10107-005-0686-0.

[3] T. Alon and N. Halman. Automatic generation of fptases for stochastic monotone dynamic programs made easier. *SIAM Journal on Discrete Mathematics*, 35(4):2679–2722, 2021. doi: 10.1137/19M1308633. URL https://doi.org/10.1137/19M1308633.

[4] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11797. URL https://ojs.aaai.org/index.php/AAAI/article/view/11797.

[5] E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999. doi: 10.1201/9781315140223.

[6] A. Badanidiyuru, R. Kleinberg, and A. Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216, 2013. doi: 10.1109/FOCS.2013.30.

[7] Q. Bai, A. Singh Bedi, and V. Aggarwal. Achieving zero constraint violation for constrained reinforcement learning via conservative natural policy gradient primal-dual algorithm. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):6737–6744, 6 2023. doi: 10.1609/aaai.v37i6.25826. URL https://ojs.aaai.org/index.php/AAAI/article/view/25826.

[8] K. Banihashem, A. Singla, and G. Radanovic. Defense against reward poisoning attacks in reinforcement learning, 2021.

[9] V. Behzadan and A. Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In P. Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, pages 262–275, Cham, 2017. Springer International Publishing. ISBN 978-3-319-62416-7.

[10] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/766ebcd59621e305170616ba3d3dac32-Paper.pdf.

[11] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Math. Oper. Res.*, 16(3):580–595, 8 1991. ISSN 0364-765X.

[12] A. Bhalgat, A. Goel, and S. Khanna. Improved approximation results for stochastic knapsack problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, page 1647–1665, USA, 2011. Society for Industrial and Applied Mathematics.

[13] A. Bhatia, P. Varakantham, and A. Kumar. Resource constrained deep reinforcement learning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 29(1):610–620, 5 2021. doi: 10.1609/icaps.v29i1.3528. URL https://ojs.aaai.org/index.php/ICAPS/article/view/3528.

[14] V. Borkar. An actor-critic algorithm for constrained markov decision processes. *Systems & Control Letters*, 54(3):207–213, 2005. ISSN 0167-6911. doi: https://doi.org/10.1016/j.sysconle.2004.08.007. URL https://www.sciencedirect.com/science/article/pii/S0167691104001276.

[15] V. Borkar and R. Jain. Risk-constrained markov decision processes. *IEEE Transactions on Automatic Control*, 59(9):2574–2579, 2014. doi: 10.1109/TAC.2014.2309262.

[16] D. M. Bossens and N. Bishop. Explicit explore, exploit, or escape (e4): Near-optimal safety-constrained reinforcement learning in polynomial time. *Mach. Learn.*, 112(3):817–858, 6 2022. ISSN 0885-6125. doi: 10.1007/s10994-022-06201-z. URL https://doi.org/10.1007/s10994-022-06201-z.

[17] K. Brantley, M. Dudík, T. Lykouris, S. Miryoosefi, M. Simchowitz, A. Slivkins, and W. Sun. Constrained episodic reinforcement learning in concave-convex and knapsack settings. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/bc6d753857fe3dd4275dff707dedf329-Abstract.html.

[18] A. Castellano, H. Min, E. Mallada, and J. A. Bazerque. Reinforcement learning with almost sure constraints. In R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, and M. Kochenderfer, editors, *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pages 559–570. PMLR, 6 2022. URL https://proceedings.mlr.press/v168/castellano22a.html.

[19] A. Charnes, W. W. Cooper, and G. H. Symonds. Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Management Science*, 4(3):235–263, 1958. doi: 10.1287/mnsc.4.3.235. URL https://doi.org/10.1287/mnsc.4.3.235.

[20] X. Chen, J. Hu, L. Li, and L. Wang. Efficient reinforcement learning in factored mdps with application to constrained rl, 2021.

[21] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3387–3395, Jul. 2019. doi: 10.1609/aaai.v33i01.33013387. URL https://ojs.aaai.org/index.php/AAAI/article/view/4213.

[22] W. C. Cheung. Regret minimization for reinforcement learning with vectorial feedback and complex objectives. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/a02ffd91ece5e7efeb46db8f10a74059-Paper.pdf.

[23] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018. URL http://jmlr.org/papers/v18/15-636.html.

[24] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/4fe5149039b52765bde64beb9f674940-Paper.pdf.

[25] V. Conitzer. On stackelberg mixed strategies. *Synthese*, 193(3):689–703, 2016. doi: 10.1007/s11229-015-0927-6. URL https://doi.org/10.1007/s11229-015-0927-6.

[26] A. Coronato, M. Naeem, G. De Pietro, and G. Paragliola. Reinforcement learning for intelligent healthcare applications: A survey. *Artificial Intelligence in Medicine*, 109:101964, 2020. ISSN 0933-3657. doi: https:

//doi.org/10.1016/j.artmed.2020.101964. URL https://www.sciencedirect.com/science/article/pii/S093336572031229X.

[27] Q. Cui and L. F. Yang. Minimax sample complexity for turn-based stochastic game. In C. de Campos and M. H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1496–1504. PMLR, 7 2021. URL https://proceedings.mlr.press/v161/cui21a.html.

[28] G. B. Dantzig. A proof of the equivalence of the programming problem and the game problem. *Activity analysis of production and allocation*, 13, 1951.

[29] C. Daskalakis, N. Golowich, and K. Zhang. The complexity of markov equilibrium in stochastic games, 2022.

[30] B. C. Dean, M. X. Goemans, and J. Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, page 395–404, USA, 2005. Society for Industrial and Applied Mathematics. ISBN 0898715857.

[31] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. R. Jovanović. Provably efficient safe exploration via primal-dual policy optimization, 2020.

[32] D. A. Dolgov and E. H. Durfee. Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In *IJCAI*, volume 19, pages 1326–1331, 2005.

[33] R. Downey and M. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer New York, 2012. ISBN 9781461205159. URL https://books.google.com/books?id=HyTjBwAAQBAJ.

[34] Y. Efroni, S. Mannor, and M. Pirotta. Exploration-Exploitation in Constrained MDPs. *arXiv e-prints*, art. arXiv:2003.02189, Mar. 2020. doi: 10.48550/arXiv.2003.02189.

[35] C. Fan, C. Zhang, A. Yahja, and A. Mostafavi. Disaster city digital twin: A vision for integrating artificial and human intelligence for disaster management. *International Journal of Information Management*, 56:102049, 2021. ISSN 0268-4012. doi: https://doi.org/10.1016/j.ijinfomgt.2019.102049. URL https://www.sciencedirect.com/science/article/pii/S0268401219302956.

[36] E. A. Feinberg. Constrained discounted markov decision processes and hamiltonian cycles. *Mathematics of Operations Research*, 25(1):130–140, 2000. doi: 10.1287/moor.25.1.130.15210. URL https://doi.org/10.1287/moor.25.1.130.15210.

[37] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, page 8550–8556. IEEE Press, 2019. doi: 10.1109/ICRA.2019.8794107. URL https://doi.org/10.1109/ICRA.2019.8794107.

[38] A. Frieze and M. Clarke. Approximation algorithms for the m-dimensional 0–1 knapsack problem: Worst-case and probabilistic analyses. *European Journal of Operational Research*, 15(1):100–109, 1984. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(84)90053-5. URL https://www.sciencedirect.com/science/article/pii/0377221784900535.

[39] J. García, Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015. URL http://jmlr.org/papers/v16/garcia15a.html.

[40] F. Geißer, G. Povéda, F. Trevizan, M. Bondouy, F. Teichteil-Königsbuch, and S. Thiébaux. Optimal and heuristic approaches for constrained flight planning under weather uncertainty. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30(1):384–393, Jun. 2020. doi: 10.1609/icaps.v30i1.6684. URL https://ojs.aaai.org/index.php/ICAPS/article/view/6684.

[41] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[42] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2014. URL https://arxiv.org/abs/1412.6572.

[43] S. Gros, M. Zanon, and A. Bemporad. Safe reinforcement learning via projection on a safe set: How to achieve optimality? *IFAC-PapersOnLine*, 53 (2):8076–8081, 2020. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2020.12.2276. URL https://www.sciencedirect.com/science/article/pii/S2405896320329360. 21st IFAC World Congress.

[44] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll. A review of safe reinforcement learning: Methods, theory and applications, 2023.

[45] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll. A review of safe reinforcement learning: Methods, theory and applications, 2024. URL https://arxiv.org/abs/2205.10330.

[46] W. Guo, X. Wu, S. Huang, and X. Xing. Adversarial policy learning in two-player competitive games. In *International Conference on Machine Learning*, pages 3910–3919. PMLR, 2021.

[47] N. Halman and G. Nannicini. Toward breaking the curse of dimensionality: An fptas for stochastic dynamic programs with multidimensional actions and scalar states. *SIAM Journal on Optimization*, 29(2):1131–1163, 2019. doi: 10.1137/18M1208423. URL https://doi.org/10.1137/18M1208423.

[48] N. Halman, D. Klabjan, C.-L. Li, J. Orlin, and D. Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. *SIAM Journal on Discrete Mathematics*, 28(4):1725–1796, 2014. doi: 10.1137/130925153. URL https://doi.org/10.1137/130925153.

[49] T. D. Hansen, P. B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *J. ACM*, 60(1), 2 2013. ISSN 0004-5411. doi: 10.1145/2432622.2432623. URL https://doi.org/10.1145/2432622.2432623.

[50] J. C. Harsanyi. Games with incomplete information played by "bayesian" players, i–iii part i. the basic model. *Management Science*, 14(3):159–182, 1967. doi: 10.1287/mnsc.14.3.159. URL https://doi.org/10.1287/mnsc.14.3.159.

[51] J. C. Harsanyi. Games with incomplete information played by "bayesian" players part ii. bayesian equilibrium points. *Management Science*, 14(5):320–334, 1968. doi: 10.1287/mnsc.14.5.320. URL https://doi.org/10.1287/mnsc.14.5.320.

[52] J. C. Harsanyi. Games with incomplete information played by 'bayesian' players, part iii. the basic probability distribution of the game. *Management Science*, 14(7):486–502, 1968. doi: 10.1287/mnsc.14.7.486. URL https://doi.org/10.1287/mnsc.14.7.486.

[53] A. HasanzadeZonuzy, A. Bura, D. Kalathil, and S. Shakkottai. Learning with safety constraints: Sample complexity of reinforcement learning for constrained mdps. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7667–7674, 5 2021. doi: 10.1609/aaai.v35i9.16937. URL https://ojs.aaai.org/index.php/AAAI/article/view/16937.

[54] B. Holmström and R. B. Myerson. Efficient and durable decision rules with incomplete information. *Econometrica*, 51(6):1799–1819, 1983. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1912117.

[55] S. Hong and B. C. Williams. An anytime algorithm for constrained stochastic shortest path problems with deterministic policies. *Artificial Intelligence*, 316:103846, 2023. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2022.103846. URL https://www.sciencedirect.com/science/article/pii/S0004370222001862.

[56] S. Hong, S. U. Lee, X. Huang, M. Khonji, R. Alyassi, and B. C. Williams. An anytime algorithm for chance constrained stochastic shortest path problems and its application to aircraft routing. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 475–481, 2021. doi: 10.1109/ICRA48506.2021.9561229.

[57] P. Hou, W. Yeoh, and P. Varakantham. Revisiting risk-sensitive mdps: New algorithms and results. *Proceedings of the International Conference on Automated Planning and Scheduling*, 24(1):136–144, 5 2014. doi: 10.1609/icaps.v24i1.13615. URL https://ojs.aaai.org/index.php/ICAPS/article/view/13615.

[58] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies, 2017. URL https://arxiv.org/abs/1702.02284.

[59] P. Jehiel, M. Meyer-ter Vehn, B. Moldovanu, and W. R. Zame. The limits of ex post implementation. *Econometrica*, 74(3):585–610, 2006. doi: https://doi.org/10.1111/j.1468-0262.2006.00675.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0262.2006.00675.x.

[60] W. Jung, M. Cho, J. Park, and Y. Sung. Quantile constrained reinforcement learning: A reinforcement learning framework constraining outage probability. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 6437–6449. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/2a07348a6a7b2c208ab5cb1ee0e78ab5-Paper-Conference.pdf.

[61] K. C. Kalagarla, R. Jain, and P. Nuzzo. A sample-efficient algorithm for episodic finite-horizon mdp with constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):8030–8037, 5 2021. doi: 10.1609/aaai.v35i9.16979. URL https://ojs.aaai.org/index.php/AAAI/article/view/16979.

[62] M. Kearns, Y. Mansour, and S. Singh. Fast planning in stochastic games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, page 309–316, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607099.

[63] M. Khonji, A. Jasour, and B. Williams. Approximability of constant-horizon constrained pomdp. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5583–5590. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/775. URL https://doi.org/10.24963/ijcai.2019/775.

[64] P. Kolesar. A markovian model for hospital admission scheduling. *Management Science*, 16(6):B384–B396, 1970. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2628725.

[65] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):805–810, 7 2010. doi: 10.1609/aaai.v24i1.7638. URL https://ojs.aaai.org/index.php/AAAI/article/view/7638.

[66] J. Kos and D. Song. Delving into adversarial attacks on deep policies, 2017. URL https://arxiv.org/abs/1705.06452.

[67] X. Y. Lee, Y. Esfandiari, K. L. Tan, and S. Sarkar. Query-based targeted action-space adversarial policies on deep reinforcement learning agents. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, ICCPS '21, page 87–97, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383530. doi: 10.1145/3450267.3450537. URL https://doi.org/10.1145/3450267.3450537.

[68] J. Letchford, L. MacDermed, V. Conitzer, R. Parr, and C. Isbell. Computing optimal strategies to commit to in stochastic games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1380–1386, 9 2021. doi: 10.1609/aaai.v26i1.8267. URL https://ojs.aaai.org/index.php/AAAI/article/view/8267.

[69] J. Li, D. Fridovich-Keil, S. Sojoudi, and C. J. Tomlin. Augmented lagrangian method for instantaneously constrained reinforcement learning problems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, page 2982–2989. IEEE Press, 2021. doi: 10.1109/CDC45484.2021.9683088. URL https://doi.org/10.1109/CDC45484.2021.9683088.

[70] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang. Deep reinforcement learning for resource management in network slicing. *IEEE Access*, 6:74429–74441, 2018. doi: 10.1109/ACCESS.2018.2881964.

[71] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 3756–3762. AAAI Press, 2017. ISBN 9780999241103.

[72] G. Liu and L. Lai. Provably efficient black-box action poisoning attacks against reinforcement learning. In M. Ranzato, A. Beygelzimer,

Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12400–12410. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/678004486c119599ed7d199f47da043a-Paper.pdf.

[73] C. Lusena, J. Goldsmith, and M. Mundhenk. Nonapproximability results for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 14:83–103, 3 2001. doi: 10.1613/jair.714. URL https://doi.org/10.1613%2Fjair.714.

[74] Y. Ma, X. Zhang, W. Sun, and J. Zhu. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 32: 14570–14580, 2019.

[75] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, page 50–56, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450346610. doi: 10.1145/3005745.3005750. URL https://doi.org/10.1145/3005745.3005750.

[76] J. McMahan. Deterministic policies for constrained reinforcement learning in polynomial time, 2024. URL https://arxiv.org/abs/2405.14183.

[77] J. McMahan and X. Zhu. Anytime-constrained multi-agent reinforcement learning, 2024. URL https://arxiv.org/abs/2410.23637.

[78] J. McMahan and X. Zhu. Anytime-constrained reinforcement learning. In S. Dasgupta, S. Mandt, and Y. Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4321–4329. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/mcmahan24a.html.

[79] J. McMahan and X. Zhu. Anytime-constrained reinforcement learning. In S. Dasgupta, S. Mandt, and Y. Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4321–4329. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/mcmahan24a.html.

[80] D. A. Melo Moreira, K. Valdivia Delgado, L. Nunes de Barros, and D. Deratani Mauá. Efficient algorithms for risk-sensitive markov decision processes with limited budget. *International Journal of Approximate Reasoning*, 139:143–165, 2021. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2021.09.003. URL https://www.sciencedirect.com/science/article/pii/S0888613X21001389.

[81] P. Menard, O. D. Domingues, A. Jonsson, E. Kaufmann, E. Leurent, and M. Valko. Fast active learning for pure exploration in reinforcement learning. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7599–7608. PMLR, 7 2021. URL https://proceedings.mlr.press/v139/menard21a.html.

[82] M. Mowbray, P. Petsagkourakis, E. del Rio-Chanona, and D. Zhang. Safe chance constrained reinforcement learning for batch process control. *Computers & Chemical Engineering*, 157:107630, 2022. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2021.107630. URL https://www.sciencedirect.com/science/article/pii/S0098135421004087.

[83] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. ISSN 0003486X. URL http://www.jstor.org/stable/1969529.

[84] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram. Chance-constrained dynamic programming with application to risk-aware robotic space exploration. *Auton. Robots*, 39(4):555–571, 12 2015. ISSN 0929-5593. doi: 10.1007/s10514-015-9467-7. URL https://doi.org/10.1007/s10514-015-9467-7.

[85] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022.

[86] X. Pan, Y. You, Z. Wang, and C. Lu. Virtual to real reinforcement learning for autonomous driving, 2017.

[87] G. Paragliola, A. Coronato, M. Naeem, and G. De Pietro. A reinforcement learning-based approach for the risk management of e-health environments: A case study. In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 711–716, 2018. doi: 10.1109/SITIS.2018.00114.

[88] P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Towards a formalization of teamwork with resource constraints. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, United States, 2004. IEEE Computer Society. Place of conference:USA.

[89] S. Paternain, L. Chamon, M. Calvo-Fullana, and A. Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c1aeb6517a1c7f33514f7ff69047e74e-Paper.pdf.

[90] H. Peng and X. Shen. Multi-agent reinforcement learning based resource management in mec- and uav-assisted vehicular networks. *IEEE Journal on Selected Areas in Communications*, 39(1):131–141, 2021. doi: 10.1109/JSAC.2020.3036962.

[91] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.

[92] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 7974–7984. PMLR, 2020.

[93] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla. Policy teaching in reinforcement learning via environment poisoning attacks. *Journal of Machine Learning Research*, 22(210):1–45, 2021.

[94] A. Rangi, H. Xu, L. Tran-Thanh, and M. Franceschetti. Understanding the limits of poisoning attacks in episodic reinforcement learning. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3394–3400. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/471. URL https://doi.org/10.24963/ijcai.2022/471. Main Track.

[95] A. Rangi, H. Xu, L. Tran-Thanh, and M. Franceschetti. Understanding the limits of poisoning attacks in episodic reinforcement learning. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3394–3400. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/471. URL https://doi.org/10.24963/ijcai.2022/471. Main Track.

[96] M. Roderick, V. Nagarajan, and Z. Kolter. Provably safe pac-mdp exploration using analogies. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1216–1224. PMLR, 4 2021. URL https://proceedings.mlr.press/v130/roderick21a.html.

[97] K. W. Ross and R. Varadarajan. Markov decision processes with sample path constraints: The communicating case. *Operations Research*, 37(5):780–790, 1989. doi: 10.1287/opre.37.5.780. URL https://doi.org/10.1287/opre.37.5.780.

[98] A. Rubinstein. The electronic mail game: Strategic behavior under "almost common knowledge". *The American Economic Review*, 79(3):385–391, 1989. ISSN 00028282. URL http://www.jstor.org/stable/1806851.

[99] A. Russo and A. Proutiere. Towards optimal attacks on reinforcement learning policies. In *2021 American Control Conference (ACC)*, pages 4561–4567, 2021. doi: 10.23919/ACC50511.2021.9483025.

[100] L. S. Shapley. Stochastic games*. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953. doi: 10.1073/pnas.39.10.1095. URL https://www.pnas.org/doi/abs/10.1073/pnas.39.10.1095.

[101] A. Sidford, M. Wang, X. Wu, L. Yang, and Y. Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/bb03e43ffe34eeb242a2ee4a4f125e56-Paper.pdf.

[102] M. Steinmetz, J. Hoffmann, and O. Buffet. Revisiting goal probability analysis in probabilistic planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 26(1):299–307, 3 2016. doi: 10.1609/icaps.v26i1.13740. URL https://ojs.aaai.org/index.php/ICAPS/article/view/13740.

[103] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu. Stealthy and efficient adversarial attacks against deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5883–5891, 4 2020. doi: 10.1609/aaai.v34i04.6047. URL https://ojs.aaai.org/index.php/AAAI/article/view/6047.

[104] Y. Sun, R. Zheng, Y. Liang, and F. Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep RL. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JM2kFbJvvI.

[105] M. A. Taleghan and T. G. Dietterich. Efficient exploration for constrained mdps. In *2018 AAAI Spring Symposium Series*, 2018.

[106] C. Tessler, Y. Efroni, and S. Mannor. Action robust reinforcement learning and applications in continuous control. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6215–6224. PMLR, 6 2019. URL https://proceedings.mlr.press/v97/tessler19a.html.

[107] G. Thomas, Y. Luo, and T. Ma. Safe reinforcement learning by imagining the near future. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information*

Processing Systems, volume 34, pages 13859–13869. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/73b277c11266681122132d024f53a75b-Paper.pdf.

[108] E. Tretschk, S. J. Oh, and M. Fritz. Sequential attacks on agents for long-term adversarial goals, 2018. URL https://arxiv.org/abs/1805.12487.

[109] Y. L. Tsai, A. Phatak, P. K. Kitanidis, and C. B. Field. Deep Reinforcement Learning for Disaster Response: Navigating the Dynamic Emergency Vehicle and Rescue Team Dispatch during a Flood. In *AGU Fall Meeting Abstracts*, volume 2019, pages NH33B–14, Dec. 2019.

[110] S. Vaswani, L. Yang, and C. Szepesvari. Near-optimal sample complexity bounds for constrained mdps. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 3110–3122. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/14a5ebc9cd2e507cd811df78c15bf5d7-Paper-Conference.pdf.

[111] N. Vlassis, M. L. Littman, and D. Barber. On the computational complexity of stochastic controller optimization in pomdps, 2012.

[112] J. von Neumann, O. Morgenstern, and A. Rubinstein. *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*. Princeton University Press, 1944. ISBN 9780691130613. URL http://www.jstor.org/stable/j.ctt1r2gkx.

[113] Y. Vorobeychik and S. Singh. Computing stackelberg equilibria in discounted stochastic games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1478–1484, Sep. 2021. doi: 10.1609/aaai.v26i1.8234. URL https://ojs.aaai.org/index.php/AAAI/article/view/8234.

[114] Y. Wang, S. S. Zhan, R. Jiao, Z. Wang, W. Jin, Z. Yang, Z. Wang, C. Huang, and Q. Zhu. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 36593–36604. PMLR, 7 2023. URL https://proceedings.mlr.press/v202/wang23as.html.

[115] H. Wei, X. Liu, and L. Ying. Triple-q: A model-free algorithm for constrained reinforcement learning with sublinear regret and zero constraint violation. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3274–3307. PMLR, 3 2022. URL https://proceedings.mlr.press/v151/wei22a.html.

[116] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, USA, 1st edition, 2011. ISBN 0521195276.

[117] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang. Uav autonomous target search based on deep reinforcement learning in complex disaster scene. *IEEE Access*, 7:117227–117245, 2019. doi: 10.1109/ACCESS.2019.2933002.

[118] F. Wu, L. Li, Z. Huang, Y. Vorobeychik, D. Zhao, and B. Li. CROP: Certifying robust policies for reinforcement learning through functional smoothing. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=HOjLHrlZhmx.

[119] Y. Wu, J. McMahan, Y. Chen, Y. Chen, X. Zhu, and Q. Xie. Minimally modifying a markov game to achieve any nash equilibrium and value. *arXiv preprint arXiv:2311.00582*, 2023.

[120] Y. Wu, J. McMahan, X. Zhu, and Q. Xie. Reward-poisoning attacks on offline multi-agent reinforcement learning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26240. URL https://doi.org/10.1609/aaai.v37i9.26240.

[121] Y. Wu, J. McMahan, X. Zhu, and Q. Xie. On faking a Nash equilibrium. *arXiv preprint arXiv:2306.08041*, 2023.

[122] Y. Wu, J. McMahan, X. Zhu, and Q. Xie. Reward poisoning attacks on offline multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10426–10434, 2023.

[123] H. Xu and S. Mannor. Probabilistic goal markov decision processes. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, page 2046–2052. AAAI Press, 2011. ISBN 9781577355151.

[124] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10639–10646, 5 2021. doi: 10.1609/aaai.v35i12.17272. URL https://ojs.aaai.org/index.php/AAAI/article/view/17272.

[125] S. Yang, S. Khuller, S. Choudhary, S. Mitra, and K. Mahadik. Correlated Stochastic Knapsack with a Submodular Objective. In S. Chechik, G. Navarro, E. Rotenberg, and G. Herman, editors, *30th Annual European Symposium*

on Algorithms (ESA 2022), volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 91:1–91:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-247-1. doi: 10.4230/LIPIcs.ESA.2022.91. URL https://drops.dagstuhl.de/opus/volltexte/2022/17029.

[126] S. X. Yu, Y. Lin, and P. Yan. Optimization models for the first arrival target distribution function in discrete time. *Journal of Mathematical Analysis and Applications*, 225(1):193–223, 1998. ISSN 0022-247X. doi: https://doi.org/10.1006/jmaa.1998.6015. URL https://www.sciencedirect.com/science/article/pii/S0022247X98960152.

[127] H. Zhang and D. C. Parkes. Value-based policy teaching with active indirect elicitation. In *AAAI*, volume 8, pages 208–214, 2008.

[128] H. Zhang, D. C. Parkes, and Y. Chen. Policy teaching through reward function learning. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 295–304, 2009.

[129] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21024–21037. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f0eb6568ea114ba6e293f903c34d7488-Paper.pdf.

[130] H. Zhang, H. Chen, D. S. Boning, and C.-J. Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=sCZbhBvqQaU.

[131] X. Zhang, Y. Ma, A. Singla, and X. Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11225–11234. PMLR, 7 2020. URL https://proceedings.mlr.press/v119/zhang20u.html.

[132] W. Zhao, T. He, R. Chen, T. Wei, and C. Liu. State-wise safe reinforcement learning: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023. ISBN 978-1-956792-03-4. doi: 10.24963/ijcai.2023/763. URL https://doi.org/10.24963/ijcai.2023/763.

[133] W. Zheng, T. Jung, and H. Lin. The stackelberg equilibrium for one-sided zero-sum partially observable stochastic games. *Automatica*, 140: 110231, 2022. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.

2022.110231. URL https://www.sciencedirect.com/science/article/pii/S0005109822000760.