Constrained MDPs with Trajectory Constraints

Martino Bernasconi
Bocconi Universiti
martino.bernasconi@unibocconi.it

Matteo Castiglioni
Politecnico di Milano
matteo.castiglioni@polimi.it

Alberto Marchesi
Politecnico di Milano
alberto.marchesi@polimi.it

Abstract

We introduce a framework that models constrained MDPs subject to *trajectory-wise safety constraints*. These constraints require that expected future costs remain below zero conditioned on any possible trajectory—*not* merely in expectation from the initial state. This captures stronger and more realistic safety guarantees, which are crucial in high-stakes applications. We characterize the structure of optimal policies, and show that Markovian policies are in general suboptimal. Moreover, we show that computing approximately-safe optimal policies is NP-hard when the number of constraints can be arbitrarily large. Despite all these challenges, we provide an exact algorithmic characterization of optimal history-dependent (non-Markovian) policies by exploiting a suitable recursive formulation of the sets of reachable reward-cost values. Then, we employ it to design a polynomial-time approximation algorithm that computes approximately safe optimal policies by dynamic programming, assuming a constant number of constraints.

1 Introduction

Constrained Markov decision processes (CMDPs) [Altman, 1999] extend classic MDPs by incorporating cost constraints. Traditionally, the literature on CMDPs requires that the *overall* expected costs obtained from the initial state onward remain below a threshold [Efroni et al., 2020, Bai et al., 2020, Stradi et al., 2024, Liu et al., 2021]. While effective in many domains, this formulation often falls short in high-stakes settings, such as autonomous driving, healthcare, and safety-critical robotics, where constraint violations along particular trajectories can be unacceptable, even if this occurs with very low probability overall. To address this limitation, some very recent works [Castellano et al., 2022, McMahan and Zhu, 2024] have proposed CMDPs settings that aim at the other extreme, by finding policies that satisfy the constraints with probability one. However, such a requirement, although reasonable in "extremely safety-critical" applications, might be overly restrictive, as many problems do *not* admit policies that satisfy such a strong condition, simply due to MDP transitions.

In this paper, we propose an intermediate model that has less stringent requirements than those of [Castellano et al., 2022, McMahan and Zhu, 2024], while being much stronger than classic CMDP models, which only constrain overall expected costs. Specifically, we require that policies attain non-positive *expected* future costs *conditioned to any possible (partial) trajectory*. Demanding that cumulative costs stay deterministically below zero along every possible trajectory is arguably overly restrictive, as it is impossible to eliminate the inherent stochasticity of transitions. Our constraint formulation strikes a balance between such a strict requirement and the standard approach in the CMDP literature, which is recovered as a special case by simply considering the empty trajectory. In particular, our model is appropriate for recoverable resource constraints, such as energy or fuel consumption, rather than irreversible failures for which pathwise guarantees may be necessary.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Next, we introduce a toy example showing the differences between the classic CMDP approach of Altman [1999], the one recently investigated by Castellano et al. [2022] and McMahan and Zhu [2024], and ours. Consider a navigation app, where the goal is to find the shortest route between two cities while consuming less than B liters per kilometer of fuel. Classic CMDPs allow to evenly balance episodes with fuel consumption $B/2$ and $3B/2$. This may be undesirable, as it results in an highly-non-smooth consumption profile. The approach of Castellano et al. [2022] and McMahan and Zhu [2024] find a policy that, with probability one, finishes the route with a consumption less than B, *if such a policy exists*. If *not*, such an approach is unable to return a policy. Instead, our approach starts each episode as in classic CMDPs, by making sure that the average consumption over the entire episode is less than B. Moreover, at some intermediate point in the trajectory, regardless of what happened so far, the policy employed still tries to maintain the future expected average consumption below B over the remaining steps, potentially leading to much more stable consumption profiles.

1.1 Original Contributions

CMDPs with trajectory-wise safety constraints introduces challenges that are fundamentally different from those in standard CMDPs. Indeed, we show that Markovian policies are suboptimal under trajectory constraints. As a result, a key challenge lies in optimizing over history-dependent policies, which involves exponentially many trajectories. We also prove that computing approximately-safe and approximately-optimal policies is NP-hard when the number of constraints is arbitrary. Nevertheless, we show that the hardness result can be circumvented for a constant number of constraints.

Then, we provide an algorithmic characterization of optimal history-dependent policies that satisfy trajectory constraints. This is based on suitable recursively-defined polyhedral sets that encode attainable reward-cost values, and it shows how to implement optimal policies "on the fly", by tracking the realized partial trajectory and solving a sequence of "local" optimization problems. Although this algorithmic approach is *not* computationally viable, it provides a fundamental building block for our main result. In particular, it provides a template to implement (non-Markovian) history-dependent policies, which do *not* generally admit a compact explicit representation.

Building on top of the aforementioned algorithmic characterization, we design an algorithm based on *dynamic programming* that implements a history-dependent policy that is optimal and satisfies the trajectory constraints up to an additive error term $\epsilon > 0$. Intuitively, our algorithm recursively constructs all the components needed to implement the desired policy by using a grid-based approximation of the polyhedral sets encoding reachable reward-cost values. We show that, when the number of constraints is constant, our algorithm runs in time polynomial in both the size of the CMDP and $1/\epsilon$.

1.2 Related Works

CMDPs, as originally formalized by Altman [1999], involve optimizing cumulative rewards subject to constraints on the expected values of other cumulative objectives (costs). While CMDPs can be solved by linear programming [Altman, 1999], a recent line of work initiated by Efroni et al. [2020] focuses on learning optimal constraint-satisfying policies. These works often employ primal-dual schemes to handle constraints satisfaction; see, *e.g.*, [Ding et al., 2022, Ghosh et al., 2022, Zheng and Ratliff, 2020, Bai et al., 2020, Stradi et al., 2024, 2025a,b,c,d,e, Liu et al., 2021, Ding and Lavaei, 2023, Wei et al., 2021].

Other forms of constraints have been proposed in the literature on CMDPs. McMahan and Zhu [2024], Castellano et al. [2022], Roderick et al. [2021] ask that all produced trajectories are safe, which in many cases might produce an arbitrarily suboptimal policy if any at all, albeit this might be a very reasonable demand in applications (*i.e.*, medical) in which errors are completely forbidden.

Our work is also related to some literature on Markov games. In these games, equilibria are defined through constraints on rewards along trajectories, which require of non-Markovian policies. MacDermed et al. [2011] and Letchford et al. [2012] focus on correlated and Stackelberg equilibria, respectively. Moreover, recent works have initiated the study of commitment problems in MDPs. Bernasconi et al. [2023] provides an algorithm to design approximately-optimal and approximately-incentive-compatible signaling policies for *Bayesian* persuasion, while Bollini et al. [2024] focuses on designing contracts. Finally, Gan et al. [2023] studies general principal-agent problems in MDPs.

Relationship to State Augmentation. Our approach is also related to methods that recover Markovian policies by augmenting the state space with additional safety-related information, such as McMahan and Zhu [2024], Sootla et al. [2022]. A fundamental distinction concerns what the augmented variable represents. These methods track cumulative past costs, whose evolution follows directly from realized costs. Our algorithm instead tracks a target for the conditional expected costs from the current step onward. Crucially, this target cannot be obtained through a simple cumulative-cost update: it depends on which future reward-cost combinations are achievable. Determining these feasible continuations and the corresponding target updates requires the backward dynamic program developed in Section 6. Thus, while our resulting policy also admits an augmented-state interpretation, constructing the relevant augmented dynamics is itself a central algorithmic challenge.

2 CMDPs With Trajectory Safety

2.1 Constrained MDPs

We study *time-inhomogeneous finite-horizon* CMDPs. These are defined by tuples

$$\mathcal{M} \coloneqq \left(\mathcal{S}, \mathcal{A}, H, m, \{p_h, r_h\}_{h\in\mathcal{H}}, \{c_{h,i}\}_{h\in\mathcal{H}, i\in[m]}, \beta\right).$$

$\mathcal{S}$ is a finite set of states. $\mathcal{A}$ is a finite set of actions. H is the time horizon with $\mathcal{H} \coloneqq [H]$ denoting the set of time steps.[1] m is the number of constraints. $p_h : \mathcal{S}\times\mathcal{A} \to \Delta(\mathcal{S})$ is a transition function for step $h \in \mathcal{H}$ with $p_h(s'|s,a)$ denoting the probability of reaching state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ at step h. $r_h : \mathcal{S}\times\mathcal{A} \to [0,1]$ is a reward function for step $h \in \mathcal{H}$. $c_{h,i} : \mathcal{S}\times\mathcal{A} \to [-1,1]$ is a cost function of constraint $i \in [m]$ at step $h \in \mathcal{H}$. Finally, $\beta \in \Delta(\mathcal{S})$ is an initial state distribution.

Given a step $h \in \mathcal{H}$ and a state $s \in \mathcal{S}$, we let $\mathcal{T}_h(s)$ be the set of all the possible (partial) *trajectories* of length h that end in state s. Formally:

$$\mathcal{T}_h(s) \coloneqq \left\{\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \mid s_h = s \wedge \forall h' \in [h-1] : s_{h'} \in \mathcal{S}, a_{h'} \in \mathcal{A}\right\}.$$

For ease of notation, we let $\mathcal{T}_h \coloneqq \bigcup_{s\in\mathcal{S}} \mathcal{T}_h(s)$ be the set of all the trajectories of length $h \in \mathcal{H}$, while $\mathcal{T} \coloneqq \bigcup_{h\in\mathcal{H}} \mathcal{T}_h$ denotes the set of all the possible trajectories (of any length). Furthermore, given a trajectory $\tau \in \mathcal{T}_h$ of length $h < H$, we denote by $\tau \oplus (a, s')$ the trajectory $\tau' \in \mathcal{T}_{h+1}$ of length $h+1$ obtained by appending action $a \in \mathcal{A}$ and state $s' \in \mathcal{S}$ to τ.

A *history-dependent policy* maps any possible trajectory to a probability distribution over actions. Formally, it is defined by a function $\pi : \mathcal{T} \to \Delta(\mathcal{A})$, with $\pi(a|\tau)$ denoting the probability of action $a \in \mathcal{A}$ when the trajectory is $\tau \in \mathcal{T}$. We denote by Π^{HD} the set of all the history-dependent policies.

We also introduce *Markovian policies*, in which action probabilities only depend on the last state rather than the whole trajectory. Formally, the set Π^{MK} of Markovian policies is defined as:

$$\Pi^{\text{MK}} \coloneqq \left\{\pi \in \Pi^{\text{HD}} \mid \pi(\tau) = \pi(\tau')\forall h \in \mathcal{H}, \forall s \in \mathcal{S}, \forall \tau, \tau' \in \mathcal{T}_h(s)\right\}.$$

Notice that $\Pi^{\text{MK}} \subseteq \Pi^{\text{HD}}$. In the following, for ease of presentation, we compactly define Markovian policies as functions $\pi : \mathcal{H}\times\mathcal{S} \to \Delta(\mathcal{A})$ and use $\pi_h(a|s)$ to denote the probability of action $a \in \mathcal{A}$ for all the trajectories of length $h \in \mathcal{H}$ whose last state is $s \in \mathcal{S}$.

2.2 Trajectory Safety and Optimal Policies

Given a history-dependent policy $\pi : \mathcal{T} \to \Delta(\mathcal{A})$ and a step $h \in \mathcal{H}$, we introduce the value function $V_h^\pi : \mathcal{T}_h \times \mathcal{A} \to \mathbb{R}$ to encode the expected rewards obtained by using π from step h onwards, for every possible trajectory of length h and action played at step h. Formally, for every $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ and $a \in \mathcal{A}$, it holds:

$$V_h^\pi(\tau, a) \coloneqq r_h(s_h, a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h, a) V_{h+1}^\pi(\tau \oplus (a, s')),$$

where, for every step $h \in \mathcal{H}$, the value function $V_h^\pi : \mathcal{T}_h \to \mathbb{R}$ is defined as

$$V_h^\pi(\tau) \coloneqq \sum_{a\in\mathcal{A}} \pi(a|\tau) V_h^\pi(\tau, a) \quad \forall \tau \in \mathcal{T}_h,$$

[1] In this paper, we denote by $[n] \coloneqq \{1, \ldots, n\}$ the set of the first $n \in \mathbb{N}_{>0}$ natural numbers.

while we let $V^{\pi}_{H+1}(\cdot) \coloneqq 0$ for ease of notation. We also introduce $V^{\pi} \coloneqq \sum_{s \in \mathcal{S}} \beta(s) V^{\pi}_1((s))$ to denote the expected reward obtained when employing a policy π from the initial state. Similarly, given a policy $\pi : \mathcal{T} \to \Delta(\mathcal{A})$, we also introduce functions encoding the $m \in \mathbb{N}_{>0}$ expected incurred costs. In particular, for every step $h \in \mathcal{H}$ and constraint $i \in [m]$, we define the constraint function $C^{\pi}_{h,i} : \mathcal{T}_h \times \mathcal{A} \to \mathbb{R}$ such that, for every $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ and $a \in \mathcal{A}$, it holds:

$$C^{\pi}_{h,i}(\tau, a) \coloneqq c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s' | s_h, a) C^{\pi}_{h+1,i}(\tau \oplus (a, s')),$$

where, for every step $h \in \mathcal{H}$, the constraint function $C^{\pi}_{h,i} : \mathcal{T}_h \to \mathbb{R}$ is defined as

$$C^{\pi}_{h,i}(\tau) \coloneqq \sum_{a \in \mathcal{A}} \pi(a|\tau) C^{\pi}_{h,i}(\tau, a) \quad \forall \tau \in \mathcal{T}_h,$$

while we let $C^{\pi}_{H+1,i}(\cdot) \coloneqq 0$ for every $i \in [m]$ for ease of notation. The value $C^{\pi}_{h,i}(\tau)$ represents the expected cost of constraint i from step h onwards after trajectory τ is realized under policy π.

We are now ready to define policies that satisfy our notion of *trajectory safety*. Intuitively, these are policies that satisfy expected cost constraints starting from every step and for every possible trajectory ending in such a step. This is a much stronger requirement than ensuring that the constraints are only satisfied for the overall expected costs obtained from the initial state up to the end of the time horizon, as done in classical CMDPs. Formally, we introduce the definition:

Definition 2.1 (Trajectory Safety)**.** Given a CMDP and an approximation error $\epsilon \geq 0$, a history-dependent policy $\pi : \mathcal{T} \to \Delta(\mathcal{A})$ is *trajectory ϵ-safe* (or simply *ϵ-safe* for short) if it satisfies the following condition:

$$C^{\pi}_{h,i}(\tau) \leq \epsilon \quad \forall i \in [m], \forall \tau \in \mathcal{T}_h.$$

If the condition above is satisfied for $\epsilon = 0$, we say that π is *trajectory safe* (or simply *safe* for short).

In the following, we denote by $\Pi^{\text{HD}}_{\epsilon} \subseteq \Pi^{\text{HD}}$ the set of all the ϵ-safe history-dependent policies. We make the minimal assumption that $\Pi^{\text{HD}}_0 \neq \varnothing$, *i.e.*, there always exists at least one safe policy. This assumption does not require realized costs to remain below the threshold almost surely, and is thus less restrictive than almost-sure feasibility of McMahan and Zhu [2024]. Nevertheless, it excludes instances containing histories from which no feasible continuation exists, even in expectation. As discussed in the introduction, our framework is intended for applications in which such recovery is possible.

Finally, we define optimal policies in CMDPs subject to trajectory safety requirements.

Definition 2.2 (Optimal Policy)**.** Given a CMDP subject to trajectory safety requirements, a policy $\pi^{\star} \in \Pi^{\text{HD}}$ is *optimal* if it is trajectory safe, namely $\pi^{\star} \in \Pi^{\text{HD}}_0$, and it achieves expected reward

$$V^{\pi^{\star}} = \text{OPT} \coloneqq \max_{\pi \in \Pi^{\text{HD}}_0} V^{\pi}.$$

Our goal is to design algorithms computing an approximation of an optimal policy. In this paper, we consider the planning problem. We leave the study of sample complexity bounds for future work.

3 Why Trajectory Safety?

Existing literature on CMDPs focuses on satisfying constraints on the overall expected costs obtained from the initial state up to the end of the time horizon. A major criticism of this approach is that it does *not* preclude significant violations of the constraints along individual trajectories. This is particularly problematic in safety-critical applications, which are arguably the primary domain of CMDPs.

In this section, we provide an example showing that our notion of trajectory safety effectively addresses the limitations of existing approaches. It is important to emphasize that requiring cumulative costs to deterministically satisfy the constraints along every possible (partial) trajectory is overly restrictive, as it is *not* possible to eliminate the inherent stochasticity due to state transitions.

We consider a CMDP with 4 states (Figure 1). The initial state is s_1, *i.e.*, $\beta(s_1) = 1$. In this state, there is only one possible action, called a_1, with $p(s_2|s_1, a_1) = \frac{1}{2}$ and $p(s_3|s_1, a_1) = \frac{1}{2}$, omitting the dependence from h since the CMDP is time-homogeneous. In state s_2, there is a single action a_2 that deterministically brings to a state s_4, while in state s_3, there is a single action a_3 that deterministically brings to s_4 as well. In state s_4, there are two actions, a_4 and a_5. The time horizon is $H = 3$; hence, after taking an action in s_4 the CMDP terminates. The rewards are defined as follows: the only action with positive reward is a_4, with reward 1, while all the other actions have reward zero. There is a single cost constraint. Action a_3 has cost 1, action a_5 has cost -1, while all the other actions have cost zero.

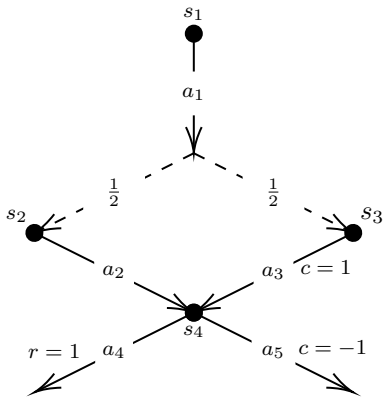

Figure 1: Dashed arrows represent transitions (notice that arrows are *not* shown when transitions are deterministic), while solid arrows denote actions. Only rewards/costs different from zero are specified.

In the example in Figure 1, a (Markovian) policy that chooses $\pi(a_4|s_4) = \pi(a_5|s_4) = \frac{1}{2}$ ensures that the overall expected cost cumulated from the initial state is equal to zero. Thus, it satisfies the constraint enforced by classical CMDPs. However, notice that such a policy incurs in a cost of 1 when the realized trajectory is $(s_1, a_1, s_3, a_3, s_4)$, and this cost is compensated when the realized trajectory is $(s_1, a_1, s_2, a_2, s_4)$, where the cost is -1. Our trajectory safety requirement rules out this behavior, since the expected cost of the policy defined above is $1 + \frac{1}{2} \cdot (-1) > 0$ when starting from the (partial) trajectory (s_1, a_1, s_3). Trajectory safety requirements can be easily met by considering a history-dependent policy that chooses $\pi(a_5|\tau) = 1$ whenever $\tau = (s_1, a_1, s_3, a_3, s_4)$, while any action for the other possible trajectory $\tau = (s_1, a_1, s_2, a_2, s_4)$. Notice that, an optimal trajectory-safe history-dependent policy should choose $\pi(a_4|\tau) = 1$ for $\tau = (s_1, a_1, s_2, a_2, s_4)$, in order to collect the only reward different from zero while satisfying trajectory constraints.

4 The Two Main Challenges of Being Trajectory Safe

Before presenting our main results, we highlight two challenges that make computing optimal policies subject to trajectory safety requirements a difficult task.

Non-Markovian Policies Are Needed The first challenge is that Markovian policies are suboptimal in CMDPs subject to trajectory safety requirements, and, thus, history-dependent policies are needed. This is formalized by the following proposition, which is proved by means of the CMDP example in Section 3 (Figure 1), in which history-dependent policies attain larger rewards than Markovian ones.

Proposition 4.1. *There exists a CMDP subject to trajectory safety requirements in which an optimal history-dependent policy gets expected rewards larger than those attainable by means of any trajectory-safe Markovian policy.*

Hardness With an Arbitrarily Large Number of Constraints The second challenge is that, if the number of cost constraints m can be arbitrarily large, then computing an approximately-optimal policy is computationally intractable even when one seeks to satisfy the trajectory constraints only approximately (*i.e.*, up to an additive error). This is made formal by the following theorem.

Theorem 4.2. *There is a function $I \mapsto \epsilon(I) := 1/\text{poly}(I)$, where I is the size of the input instance and* $\text{poly}(I)$ *is a suitable polynomial, such that it is* NP*-hard to find an $\epsilon(I)$-optimal history-dependent policy that is $\epsilon(I)$-trajectory safe, i.e., a policy $\pi \in \Pi^{\text{HD}}_{\epsilon(I)}$ such that it holds $V^\pi \geq \text{OPT} - \epsilon(I)$.*

Given the intractability result above, in the rest of the paper we focus on designing polynomial-time algorithms for settings in which the number of constraints m is constant. As it will become clear in the following, this task raises considerable computational challenges.

5 An Algorithmic Characterization of Optimal Policies

In this section, we provide a characterization of an optimal trajectory-safe history-dependent policy, in terms of an algorithm implementing one such policy. Such an algorithm is *not* computationally viable, but it provides the first building block towards our main result in the following Section 6. The algorithm works by first solving a recursively-defined optimization problem to find the expected reward and costs of an optimal policy. Then, the algorithm exploits such information to implement an optimal policy "on the fly", while interacting with the CMDP. Notice that, in this way, we do *not* explicitly compute all the (exponentially-many) components of an optimal history-dependent policy, but only those related to the trajectories realized while interacting with the CMDP.

5.1 A Useful Recursively-Defined Optimization Problem

For every $h \in \mathcal{H}$ and $s \in \mathcal{S}$, we introduce a set $\mathcal{G}_{h,s} \subseteq \mathbb{R}^{1+m+|\mathcal{A}|+(1+m)\cdot|\mathcal{A}|\cdot|\mathcal{S}|}$ to characterize all the possible expected rewards and costs achievable from step h onward when in state s. In particular, each element of $\mathcal{G}_{h,s}$ is a tuple $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$, where: v is the expected reward from step h onward; $\boldsymbol{c} \coloneqq \{c_i\}_{i\in[m]}$, with c_i being the expected cost of constraint i from step h onward; $\boldsymbol{g} \coloneqq \{g_a\}_{a\in\mathcal{A}}$, with g_a being the probability of playing action a in state s; $\boldsymbol{z} \coloneqq \{z_{a,s'}\}_{a\in\mathcal{A},s'\in\mathcal{S}}$, with $z_{a,s'}$ encoding the expected reward from step $h+1$ onward after playing action a and reaching state s'; $\boldsymbol{w} \coloneqq \{w_{i,a,s'}\}_{i\in[m],a\in\mathcal{A},s'\in\mathcal{S}}$, with $w_{i,a,s'}$ encoding the expected cost of constraint i from step $h+1$ onward after playing action a and reaching s'.

For every step $h \in \mathcal{H}$ and state $s \in \mathcal{S}$, given all the sets associated with the following step $h+1$, namely the sets $\mathcal{G}_{h+1,s'}$ for $s' \in \mathcal{S}$, we recursively define $\mathcal{G}_{h,s}$ as the set of all the tuples $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ that satisfy the following constraints:

$$v = \sum_{a\in\mathcal{A}} g_a \left[r_h(s,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s,a) z_{a,s'} \right] \tag{1a}$$

$$c_i = \sum_{a\in\mathcal{A}} g_a \left[c_{i,h}(s,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s,a) w_{i,a,s'} \right] \qquad \forall i \in [m] \tag{1b}$$

$$(z_{a,s'}, \boldsymbol{w}_{a,s'}) \in \text{Next}_{v,\boldsymbol{c}}(\mathcal{G}_{h+1,s'}) \qquad \forall a \in \mathcal{A}, \forall s' \in \mathcal{S} \tag{1c}$$

$$c_i \leq 0 \qquad \forall i \in [m] \tag{1d}$$

$$\sum_{a\in\mathcal{A}} g_a = 1 \tag{1e}$$

$$g_a \geq 0 \qquad \forall a \in \mathcal{A}, \tag{1f}$$

where we defined $\boldsymbol{w}_{a,s'} \coloneqq \{w_{i,a,s'}\}_{i\in[m]}$ for ease of notation. Moreover, we let $\text{Next}_{v,\boldsymbol{c}}(\mathcal{G}_{h+1,s'})$ be the projection of $\mathcal{G}_{h+1,s'}$ on the first two components, *i.e.*, the future reward and costs achievable from the following step $h+1$, when the next state is s'. Formally, we deifne $\text{Next}_{v,\boldsymbol{c}}(\mathcal{G}_{h+1,s'}) \coloneqq \{(v,\boldsymbol{c}) \mid (v,\boldsymbol{c},\boldsymbol{g},\boldsymbol{z},\boldsymbol{w}) \in \mathcal{G}_{h+1,s'}\}$. As a base case for the inductive definition above, we also artificially introduce the sets $\mathcal{G}_{H+1,s'} \coloneqq (0, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$ for all $s' \in \mathcal{S}$.

Constraints (1a) and (1b) ensure that v and $\boldsymbol{c}$, respectively, are properly defined, Constraints (1c) guarantee that $\boldsymbol{z}$ and $\boldsymbol{w}$ correctly encode the expected reward and costs, respectively, from the following step $h+1$ onward, while Constraints (1d) are those guaranteeing trajectory safety.

The recursively-defined sets $\mathcal{G}_{h,s}$ allow us to define the optimization problem solved by the algorithm implementing an optimal history-dependent policy, which we describe in more details in the following Section 5.2. Specifically, whenever the initial state is $s \in \mathcal{S}$, the algorithm maximizes the value of v over all the pairs $(v, \boldsymbol{c}) \in \mathcal{G}_{1,s}$, so as to find the expected reward and costs associated with an optimal policy. Intuitively, the recursive definition of $\mathcal{G}_{1,s}$ ensures that the value being maximized actually encodes the expected reward from the initial state s onward (Constraint (1a)) and that the maximization domain correctly captures the set Π_0^{HD} of all the history-dependent policies that are trajectory safe (Constraints (1b) and (1d)).

Notice that $\mathcal{G}_{h,s}$ cannot be computed efficiently. First, Problem 2 is quadratic. Second, the recursive definition of $\mathcal{G}_{h,s}$ cannot be "unrolled" efficiently, since the size of the "unrolled" constraints defining $\mathcal{G}_{h,s}$ would be exponential in H, including a different variable for every possible trajectory starting

from s at step h. Section 6 is devoted to the design of a computationally amenable approximation of Problem (2). The key idea is that, while there may be several trajectories that reach the same state, many can be "approximately" bucketed together.

5.2 Implementing an Optimal History-Dependent Policy

Next, we show how the definition of the sets $\mathcal{G}_{h,s}$ can be exploited to implement an optimal history-dependent policy in a CMDP subject to trajectory safety.

The procedure is detailed in Algorithm 1.

Algorithm 1 Implement Optimal Safe Policy

Require: CMDP $\mathcal{M}$

Observe initial state $s_1 \sim \beta$
$\tau \leftarrow (s_1)$ {▷ current trajectory}
$(v^\tau, \boldsymbol{c}^\tau) \leftarrow \arg\max_{(v,\boldsymbol{c}) \in \mathcal{G}_{1,s_1}} v$
for $h = 1, \dots, H$ {▷ top-down order} **do**
  $(\boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w}) \leftarrow$ Any tuple $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w}) \in \mathcal{G}_{h,s_h}$ with $v = v^\tau$ and $\boldsymbol{c} = \boldsymbol{c}^\tau$
  Play action $a_h \sim \boldsymbol{g}$
  Observe $s_{h+1} \sim p_h(\cdot|s_h, a_h)$
  $\tau \leftarrow \tau \oplus (a_h, s_{h+1})$ {▷ update trajectory}
  $(v^\tau, \boldsymbol{c}^\tau) \leftarrow (z_{a_h,s_{h+1}}, \boldsymbol{w}_{a_h,s_{h+1}})$
end for

Algorithm 1 keeps track of the trajectory realized during its execution by means of a variable τ, which is initialized to the initial state s_1 sampled from β. As a first step, the algorithm computes the expected reward and costs of an optimal policy, stored in the pair $(v^\tau, \boldsymbol{c}^\tau)$ for $\tau = (s_1)$. This is done by looking for the maximum value of v over the pairs $(v, \boldsymbol{c})$ belonging to the set $\mathcal{G}_{1,s_1}$. This information is sufficient to start implementing an optimal history-dependent policy while interacting with the CMDP. In particular, at each step $h \in \mathcal{H}$, given the trajectory $\tau = (s_1, a_1, \dots, s_{h-1}, a_{h-1}, s_h)$ of length h realized up to that point, the algorithm computes a probability distribution over actions that allows to attain the expected reward and costs $(v^\tau, \boldsymbol{c}^\tau)$ of an optimal policy from step h onwards, as well as the expected reward and costs that must be guaranteed by an optimal policy from the following step $h+1$. These components are obtained by taking any element $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ of $\mathcal{G}_{h,s_h}$ such that $v = v^\tau$ and $\boldsymbol{c} = \boldsymbol{c}^\tau$. In particular, the probability distribution $\boldsymbol{g}$ is the one employed by the algorithm to draw an action a_h to be played at step h, while, after observing the next state s_{h+1} and updating the trajectory τ to $\tau \oplus (a, s')$, the expected reward and costs $(v^\tau, \boldsymbol{c}^\tau)$ for the following step $h+1$ are defined by the pair $(z_{a_h,s_{h+1}}, \boldsymbol{w}_{a_h,s_{h+1}})$.

Next, we show that the history-dependent policy implemented by Algorithm 1 is indeed optimal. This amounts to showing the following results:

Lemma 5.1. *Given a CMDP $\mathcal{M}$, the history-dependent policy implemented by Algorithm 1 is safe.*

In particular, the proof of Lemma 5.1 is based on induction and proves that any trajectory τ generated by Algorithm 1 satisfies its corresponding trajectory-wise constraint in Definition 2.1.

Similarly, the next lemma shows that the policy implemented by Algorithm 1 is also optimal.

Lemma 5.2. *Given a CMDP $\mathcal{M}$, the history-dependent policy $\pi : \mathcal{T} \to \Delta(\mathcal{A})$ implemented by Algorithm 1 is such that $V^\pi = \text{OPT}$.*

The two results above give us the following theorem:

Theorem 5.3. *Given a CMDP $\mathcal{M}$ subject to trajectory safety requirements, the history-dependent policy implemented by Algorithm 1 is optimal.*

Algorithm 1, together with Theorem 5.3, provides an algorithmic characterization of an optimal policy in CMDPs subject to trajectory safety requirements.

Algorithm 1 is *not* a viable computational tool to actually implement such a policy. Indeed, it involves a crucial step that cannot be implemented efficiently, namely, solving the optimization problem over the recursively-defined sets $\mathcal{G}_{h,s}$ in order to compute the expected reward and costs of an optimal policy. The following section is devoted to the design of a computationally efficient approach.

6 Implementing Efficiently an Approximately-Safe Optimal Policy

We design an algorithm that allows to efficiently implement an approximately-safe optimal policy in CMDPs subject to trajectory safety requirements.

Given an approximation error $\epsilon > 0$, our algorithm implements an history-dependent policy $\pi \in \Pi^{\text{HD}}$ that is trajectory ϵ-safe, namely $\pi \in \Pi^{\text{HD}}_{\epsilon}$, and it achieves an expected reward $V^{\pi} \geq \text{OPT}$. Moreover, we show that our algorithm is computationally efficient. In particular, when the number of constraints m is a constant, the algorithm runs in time polynomial in $1/\epsilon$ and the size of the CMDP given as input.

Our algorithm works by leveraging a suitable discretization of expected costs, consisting in a finite grid that approximates the hypercube $[-H, 0]^m$, which is the set of possible non-positive expected costs. In particular, given a suitably-defined step size parameter $\delta \in (0, 1)$, the algorithm considers discretized expected costs $\tilde{\boldsymbol{c}} \coloneqq \{\tilde{c}_i\}_{i \in [m]}$ belonging to the set $\mathcal{D}^m_\delta$, where

$$\mathcal{D}_\delta \coloneqq \{-k\delta \mid k \in \mathbb{N}_0 \wedge k \leq \lceil H/\delta \rceil\}.$$

Notice that $|\mathcal{D}_\delta| = \lceil H/\delta \rceil + 1$, zero is always contained in $\mathcal{D}_\delta$, and the lowest possible discretized expected cost in $\mathcal{D}_\delta$ is greater than or equal to $-(H + \delta)$.

For ease of notation, given $\delta \in (0, 1)$, we use $\lfloor \cdot \rfloor_\delta$ to denote a rounding operator mapping numbers to the nearest multiple of δ *below* them. Specifically, given any real number $x \in \mathbb{R}$, we let $\lfloor x \rfloor_\delta \coloneqq \max\{k\delta \mid k \in \mathbb{Z} \wedge k\delta \leq x\}$. Similarly, $\lceil \cdot \rceil_\delta$ rounds numbers to the nearest multiples of δ *above* them, with $\lceil x \rceil_\delta \coloneqq \min\{k\delta \mid k \in \mathbb{Z} \wedge k\delta \geq x\}$. We also extend the operators $\lfloor \cdot \rfloor_\delta$ and $\lceil \cdot \rceil_\delta$ to work with vectors of real numbers, with the rounding operation being applied to the vector component wise.

Our algorithm, whose pseudocode is detailed in Algorithm 3, efficiently implements an approximately-safe optimal policy $\pi \in \Pi^{\text{HD}}$ by using a procedure that follows the line of Algorithm 1, by using discretized expected costs instead of "exact" ones. Crucially, using discretized expected costs allows to tackle the optimization problem in Line 3 of Algorithm 1 by means of an *ad hoc* dynamic programming procedure, whose pseudocode is in Algorithm 2.

6.1 Dynamic Programming Procedure

The dynamic programming procedure in Algorithm 2 works in a *bottom-up* fashion, from the last step backwards, in order to compute all the components needed to implement an approximately-safe optimal policy. These components are stored in suitable tables M_h, G_h, and W_h, indexed by the step $h \in \mathcal{H}$ they refer to.

Intuitively, the table entries $M_h(s, \tilde{\boldsymbol{c}})$, $G_h(s, \tilde{\boldsymbol{c}})$, and $W_h(s, \tilde{\boldsymbol{c}}, a, s')$ are used to encode an approximate solution to the optimization problem over the sets $\mathcal{G}_{h,s}$ defined by Equation (1) (see Line 3 in Algorithm 1).

Specifically, $M_h(s, \tilde{\boldsymbol{c}})$ is the value of an optimal policy starting from state s at step h, subject to the constraint that expected costs remain approximately below $\tilde{\boldsymbol{c}}$ from that point onward. Moreover, $G_h(s, \tilde{\boldsymbol{c}})$ and $W_h(s, \tilde{\boldsymbol{c}}, a, s')$ contain the action-selection strategy and the discretized future expected costs, respectively, for such a policy. For each step $h \in \mathcal{H}$ (in bottom-up order), state $s \in \mathcal{S}$, and discretized expected costs $\tilde{\boldsymbol{c}} \in \mathcal{D}^m_\delta$, Algorithm 2 computes an optimal action-selection policy subject to the constraint that expected costs remain approximately below $\tilde{\boldsymbol{c}}$ from that point onward, by using the information already computed for the following step $h + 1$. This is done by solving problem $\mathcal{P}^\delta_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ in Line 9. This is parametrized by some discretized expected costs $\tilde{\boldsymbol{c}} \in \mathcal{D}^m_\delta$ and a set-valued function $\mathcal{F}_{h+1}$, which is recursively defined in a bottom-up fashion in Line 16. The latter

Algorithm 2 Dynamic Programming

Require: CMDP $\mathcal{M}$, step size $\delta \in (0, 1)$
1: Initialize $M_h(s, \tilde{\boldsymbol{c}}) \leftarrow \bot$ for $h \in \mathcal{H}, s \in \mathcal{S}, \tilde{\boldsymbol{c}} \in \mathcal{D}^m_\delta$
2: Initialize $G_h(s, \tilde{\boldsymbol{c}}) \leftarrow \bot$ for $h \in \mathcal{H}, s \in \mathcal{S}, \tilde{\boldsymbol{c}} \in \mathcal{D}^m_\delta$
3: Initialize $W_h(s, \tilde{\boldsymbol{c}}, a, s') \leftarrow \bot$ for all $h \in \mathcal{H}, s \in \mathcal{S}, \tilde{\boldsymbol{c}} \in \mathcal{D}^m_\delta, a \in \mathcal{A}, s' \in \mathcal{S}$
4: Initialize $\mathcal{F}_{H+1}(s) \leftarrow (0, \mathbf{0})$ for all $s \in \mathcal{S}$
5: **for** $h = H, \ldots, 1$ {▷ bottom-up order} **do**
6: **for** $s \in \mathcal{S}$ **do**
7: **for** $\tilde{\boldsymbol{c}} \in \mathcal{D}^m_\delta$ **do**
8: **if** problem $\mathcal{P}^\delta_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ is *feasible* **then**
9: $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w}) \leftarrow$ Optimal solution to problem $\mathcal{P}^\delta_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$
10: $M_h(s, \tilde{\boldsymbol{c}}) \leftarrow v$
11: $G_h(s, \tilde{\boldsymbol{c}}) \leftarrow \boldsymbol{g}$
12: $W_h(s, \tilde{\boldsymbol{c}}, a, s') \leftarrow \lceil \boldsymbol{w}_{a,s'} \rceil_\delta \; \forall a \in \mathcal{A}, s' \in \mathcal{S}$
13: **end if**
14: **end for**
15: $\mathcal{V} \leftarrow \{(v, \tilde{\boldsymbol{c}}) \mid \tilde{\boldsymbol{c}} \in \mathcal{D}^m_\delta \wedge v = M_h(s, \tilde{\boldsymbol{c}}) \neq \bot\}$
16: $\mathcal{F}_h(s) \leftarrow$ H-representation of polytope co $(\mathcal{V})$
17: **end for**
18: **end for**
19: **return** $\{M_h, G_h, W_h\}_{h \in \mathcal{H}}$

maps every state $s \in \mathcal{S}$ to a polytope $\mathcal{F}_{h+1}(s)$, which is constructed by Algorithm 2 at step $h+1$ and is defined as:

$$\mathcal{F}_{h+1}(s) \coloneqq \left\{ (z, \boldsymbol{w}) \mid A_{h+1,s} \begin{pmatrix} z \\ \boldsymbol{w} \end{pmatrix} \leq b_{h+1,s} \right\},$$

where $A_{h+1,s} \in \mathbb{R}^{d \times (1+m)}$ and $b_{h+1,s} \in \mathbb{R}^d$ are suitably-defined matrices and vectors, respectively (for a suitable dimension $d \in \mathbb{N}$). In particular, since $\mathcal{V}$ is a finite set of polynomial size, it is possible to represent its convex hull with a polynomial set of hyperplanes, *i.e.*, linear inequalities (see the proof of Theorem 6.5 for additional details). Problem $\mathcal{P}^{\delta}_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ is formally defined as follows:

$$\max_{v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w}} \quad v \quad \text{s.t.} \tag{2a}$$

$$v = \sum_{a \in \mathcal{A}} g_a \left[r_h(s,a) + \sum_{s' \in \mathcal{S}} p_h(s'|s,a) z_{a,s'} \right] \tag{2b}$$

$$c_i = \sum_{a \in \mathcal{A}} g_a \left[c_{i,h}(s,a) + \sum_{s' \in \mathcal{S}} p_h(s'|s,a) w_{i,a,s'} \right] \qquad \forall i \in [m] \tag{2c}$$

$$(z_{a,s'}, \boldsymbol{w}_{a,s'}) \in \mathcal{F}_{h+1}(s') \qquad \forall a \in \mathcal{A}, \forall s' \in \mathcal{S} \tag{2d}$$

$$c_i \leq \tilde{c}_i + \delta \qquad \forall i \in [m] \tag{2e}$$

$$\sum_{a \in \mathcal{A}} g_a = 1 \tag{2f}$$

$$g_a \geq 0 \qquad \forall a \in \mathcal{A}, \tag{2g}$$

where the variables $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ have the same interpretation as in Equation (1). The problem maximizes the reward v subject to constraints resembling those in Equation (1). The only differences are that (i) expected costs are required to approximately stay (up to error δ) below the discretized value $\tilde{\boldsymbol{c}}$ and (ii) the set in Equation (1c) is replaced by its approximation $\mathcal{F}_{h+1}(s')$. After Algorithm 2 has solved problem $\mathcal{P}^{\delta}_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ in Line 9, the results are stored in the table entries $M_h(s, \tilde{\boldsymbol{c}})$, $G_h(s, \tilde{\boldsymbol{c}})$, and $W_h(s, \tilde{\boldsymbol{c}}, a, s')$. Specifically, the entry $M_h(s, \tilde{\boldsymbol{c}})$ is set to the value of the expected reward v, the entry $G_h(s, \tilde{\boldsymbol{c}})$ is set to contain the probability distribution over actions $\boldsymbol{g}$, while each entry $W_h(s, \tilde{\boldsymbol{c}}, a, s')$ is set to the distretized version of the future expected costs $\boldsymbol{w}_{a,s'}$ when state s' is reached after action a. Crucially, $G_h(s, \tilde{\boldsymbol{c}})$ and $W_h(s, \tilde{\boldsymbol{c}}, a, s')$ are the fundamental components that allow to implement the approximately-safe optimal policy computed by dynamic programming.

6.2 Implementation Procedure

Finally, Algorithm 3 implements an approximately-safe optimal policy by using the elements computed by means of Algorithm 2. In particular, Algorithm 3 works exactly as Algorithm 1, but instead of using the elements in the exact sets $\mathcal{G}_{h,s}$, it uses their discretized values stored in the tables G_h and W_h. Notice that the table M_h is only used to store the reward values, and, thus, it is *not* actually used during execution, but only for the internal working of Algorithm 2.

Algorithm 3 Implement Approximate Policy

Require: CMDP $\mathcal{M}$, error $\epsilon > 0$

1: $\delta \leftarrow \min\{\epsilon/2H, 1/2\}$
2: $\{M_h, G_h, W_h\}_{h \in \mathcal{H}} \leftarrow$ Run Algorithm 2 with $\mathcal{M}$ and δ as inputs
3: Observe initial state $s_1 \sim \beta$
4: $\tau \leftarrow (s_1)$ {▷ current trajectory}
5: $\tilde{\boldsymbol{c}}^{\tau} \leftarrow \mathbf{0}$ {▷ current discretized expected costs}
6: **for** $h = 1, \ldots, H$ {▷ top-down order} **do**
7: $\quad \boldsymbol{g} \leftarrow G_h(s_h, \tilde{\boldsymbol{c}}^{\tau})$
8: $\quad$ Play action $a_h \sim \boldsymbol{g}$
9: $\quad$ Observe $s_{h+1} \sim p_h(\cdot|s_h, a_h)$
10: $\quad \tilde{\boldsymbol{c}}^{\tau \oplus (a_h, s_{h+1})} \leftarrow W_h(s_h, \tilde{\boldsymbol{c}}^{\tau}, a_h, s_{h+1})$ {▷ compute $\tilde{c}^{\tau}$ for the next step $h+1$}
11: $\quad \tau \leftarrow \tau \oplus (a_h, s_{h+1})$ {▷ update trajectory}
12: **end for**

6.3 Theoretical Guarantees

We conclude by proving the theoretical guarantees provided by Algorithm 3. First, we show that the policy implemented by Algorithm 3 guarantees an expected reward larger than the "approximated ones" stored in M_h. Formally:

Lemma 6.1. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$, let $\pi : \mathcal{T} \times \mathcal{A} \to [0,1]$ be the history-dependent policy implemented by Algorithm 3. Then, for every step $h \in \mathcal{H}$ and trajectory $\tau = (s_1, a_1, \dots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h, it holds that $V_h^{\pi}(\tau) \geq M_h(s_h, \tilde{\boldsymbol{c}}^{\tau})$, where $\tilde{\boldsymbol{c}}^{\tau}$ is defined in Algorithm 3.*

The lemma shows that the implemented policy gets an expected reward larger than the one in M_h. The next natural step is to showing that M_h has a larger expected reward than an optimal policy $\pi^{\star}$:

Lemma 6.2. *Let $\pi^{\star} : \mathcal{T} \times \mathcal{A} \to [0,1]$ be an optimal history-dependent policy in a CMDP $\mathcal{M}$ subject to trajectory safety requirements. Then, given $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, Algorithm 3 ensures that $M_h(s_h, \lfloor \boldsymbol{c}^{\star,\tau} \rfloor_{\delta}) \geq V_h^{\pi^{\star}}(\tau)$ for every step $h \in \mathcal{H}$ and trajectory $\tau = (s_1, a_1, \dots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h, where δ is defined in Algorithm 3 and we let $\boldsymbol{c}^{\star,\tau} \coloneqq \{c_i^{\star,\tau}\}$ with $c_i^{\star,\tau} \coloneqq C_{h,i}^{\pi^{\star}}(\tau)$.*

Finally, our last lemma shows that the policy implemented by Algorithm 3 is ϵ-safe. This follows from the observation that Algorithm 2 is approximating the expected costs up to an additive error related to δ. Hence, for δ small enough, the cumulative error is at most ϵ. Formally:

Lemma 6.3. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, the history-dependent policy implemented by Algorithm 3 is trajectory ϵ-safe.*

Now, by combining the previous results, we directly derive the guarantees provided by Algorithm 3.

Theorem 6.4. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, the history-dependent policy $\pi : \mathcal{T} \times \mathcal{A} \to [0,1]$ implemented by Algorithm 3 is trajectory ϵ-safe and achieves $V^{\pi} \geq \text{OPT}$.*

To conclude, we need to bound the running time of Algorithm 3. This amounts to design oracles that implement Lines 9 and 16. The quadratic program in Lines 9 can be solved efficiently through a linear relaxation (see the proof of Theorem 6.5 for additional details), while the program in Line 16 can be solved efficiently given an efficient representation of the convex hull through linear inequalities.

Theorem 6.5. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, Algorithm 3 runs in time polynomial in $1/\epsilon$ and the size of the CMDP $\mathcal{M}$, when the number of constraints m is constant.*

Acknowledgments

Alberto Marchesi was supported by the contribution of Ministero dell'Università e della ricerca pursuant to D.D. n. 18010 of 12 November 2025 – BANDO FIS 3, Project FIS-2024-05736 (Starting Grant), title: "Towards a trustworthy strategic use of data in machine learning pipelines" (STRATDATA). CUP: D53C25002380001.

Martino Bernasconi was supported by an ERC grant (Project 101165466 — PLA-STEER).

Matteo Castiglioni was supported by the EU Horizon project ELIAS (European Lighthouse of AI for Sustainability, No. 101120237).

References

E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

Qinbo Bai, Vaneet Aggarwal, and Ather Gattami. Provably efficient model-free algorithm for MDPs with peak constraints. *arXiv preprint arXiv:2003.05555*, 2020.

Martino Bernasconi, Matteo Castiglioni, Alberto Marchesi, and Mirco Mutti. Persuading farsighted receivers in MDPs: the power of honesty. *Advances in Neural Information Processing Systems*, 36: 14987–15014, 2023.

Matteo Bollini, Francesco Bacchiocchi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Contracting with a reinforcement learning agent by playing trick or treat. *arXiv preprint arXiv:2410.13520*, 2024.

Agustin Castellano, Hancheng Min, Enrique Mallada, and Juan Andrés Bazerque. Reinforcement learning with almost sure constraints. In *Learning for Dynamics and Control Conference*, pages 559–570. PMLR, 2022.

Dongsheng Ding, Kaiqing Zhang, Jiali Duan, Tamer Başar, and Mihailo R Jovanović. Convergence and sample complexity of natural policy gradient primal-dual methods for constrained MDPs. *arXiv preprint arXiv:2206.02346*, 2022.

Yuhao Ding and Javad Lavaei. Provably efficient primal-dual reinforcement learning for CMDPs with non-stationary objectives and constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7396–7404, 2023.

Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-exploitation in constrained MDPs, 2020.

Jiarui Gan, Rupak Majumdar, Debmalya Mandal, and Goran Radanovic. Stochastic principal-agent problems: Efficient computation and learning. *arXiv preprint arXiv:2306.03832*, 2023.

Arnob Ghosh, Xingyu Zhou, and Ness Shroff. Provably efficient model-free constrained rl with linear function approximation. *Advances in Neural Information Processing Systems*, 35:13303–13315, 2022.

Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

Joshua Letchford, Liam MacDermed, Vincent Conitzer, Ronald Parr, and Charles Isbell. Computing optimal strategies to commit to in stochastic games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1380–1386, 2012.

Tao Liu, Ruida Zhou, Dileep Kalathil, Panganamala Kumar, and Chao Tian. Learning policies with zero or bounded constraint violation for constrained MDPs. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Liam MacDermed, Karthik Narayan, Charles Isbell, and Lora Weiss. Quick polytope approximation of all correlated equilibria in stochastic games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 707–712, 2011.

Jeremy McMahan and Xiaojin Zhu. Anytime-constrained reinforcement learning. In *International conference on artificial intelligence and statistics*, pages 4321–4329. PMLR, 2024.

Melrose Roderick, Vaishnavh Nagarajan, and Zico Kolter. Provably safe PAC-MDP exploration using analogies. In *International Conference on Artificial Intelligence and Statistics*, pages 1216–1224. PMLR, 2021.

Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H Mguni, Jun Wang, and Haitham Ammar. Sauté rl: Almost surely safe reinforcement learning using state augmentation. In *International Conference on Machine Learning*, pages 20423–20443. PMLR, 2022.

Francesco Emanuele Stradi, Jacopo Germano, Gianmarco Genalti, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Online learning in CMDPs: Handling stochastic and adversarial constraints. In *Forty-first International Conference on Machine Learning*, 2024.

Francesco Emanuele Stradi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Learning adversarial MDPs with stochastic hard constraints. In *Forty-second International Conference on Machine Learning*, 2025a.

Francesco Emanuele Stradi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Optimal strong regret and violation in constrained MDPs via policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Francesco Emanuele Stradi, Eleonora Fidelia Chiefari, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Beyond slater's condition in online CMDPs with stochastic and adversarial constraints, 2025c. URL https://arxiv.org/abs/2509.20114.

Francesco Emanuele Stradi, Anna Lunghi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Policy optimization for CMDPs with bandit feedback: Learning stochastic and adversarial constraints. In *Forty-second International Conference on Machine Learning*, 2025d.

Francesco Emanuele Stradi, Anna Lunghi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Taming adversarial constraints in CMDPs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025e.

Honghao Wei, Xin Liu, and Lei Ying. A provably-efficient model-free algorithm for constrained markov decision processes, 2021. URL https://arxiv.org/abs/2106.01577.

Liyuan Zheng and Lillian Ratliff. Constrained upper confidence reinforcement learning. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 620–629. PMLR, 10–11 Jun 2020.

Günter M Ziegler. Lectures on polytopes, volume 152 of. *Graduate Texts in Mathematics*, pages 6–2, 1995.

A Proofs Omitted From Section 4

Proposition 4.1. *There exists a CMDP subject to trajectory safety requirements in which an optimal history-dependent policy gets expected rewards larger than those attainable by means of any trajectory-safe Markovian policy.*

Proof. Consider the CMDP in Figure 1. We show that non-Markovian policies extract a larger utility than Markovian ones. In the following, we omit the dependence from the time step h, as this is clear from context.

Optimal Markovian Policy. Consider a Markovian policy $\pi \in \Pi^{\text{MK}}$. Notice that the only state in which more than one action is available is s_4. Hence, the policy simply sets $\pi(a_4|s_4)$ and $\pi(a_5|s_4)$. It is easy to see that the constraint $C^\pi(\tau) \leq 0$ for $\tau = (s_1, a_1, s_3)$ implies that it must be $\pi(a_5|s_4) = 1$. Hence, the expected reward of every safe Markovian policy is zero.

Optimal History-Dependent Policy. Now, consider a history-dependent policy $\pi \in \Pi^{\text{HD}}$ that sets $\pi(a_4|\tau) = 1$ for $\tau = (s_1, a_1, s_2, a_2, s_4)$ and $\pi(a_5|\tau) = 1$ for $\tau = (s_1, a_1, s_3, a_3, s_4)$, *i.e*, a policy that selects a_4 if the trajectory contains state s_2, while it chooses a_5 if the trajectory contains s_3. It is easy to see that this policy satisfies all the trajectory constraints and has expected reward $\frac{1}{2}$. □

Theorem 4.2. *There is a function $I \mapsto \epsilon(I) := 1/\text{poly}(I)$, where I is the size of the input instance and* $\text{poly}(I)$ *is a suitable polynomial, such that it is* NP*-hard to find an $\epsilon(I)$-optimal history-dependent policy that is $\epsilon(I)$-trajectory safe, i.e., a policy $\pi \in \Pi^{\text{HD}}_{\epsilon(I)}$ such that it holds $V^\pi \geq \text{OPT} - \epsilon(I)$.*

Proof. We reduce from 3-SAT. An instance of 3-SAT includes a set of variables $N = \{1, \dots, n\}$ and a set Q of clauses. Each clause q is of the form $l_1 \vee l_2 \vee l_3$, where l_i are literals (*i.e.*, variables or their negation). Then, given a set of variables N and a set of clauses Q, it is well known that the problem of deciding whether there exists an assignment that satisfies all the clauses is NP-hard [Karp, 1972].

We show that, if the 3-SAT instance admits a feasible assignment, *i.e.*, one satisfying all the clauses, then there exists a safe policy with expected reward 1, otherwise, if the instance does *not* admit any feasible assignment, then all the ϵ-safe policies achieve expected reward at most $2n\epsilon$. This is sufficient to prove the statement of the theorem.

Construction. Given a 3-SAT instance, we build an instance of our problem with states s_i, $s_{i,T}$, $s_{i,F}$, $\bar{s}_i$, $\bar{s}_{i,T}$, $\bar{s}_{i,F}$ for any $i \in N$. Moreover, we include two additional states $\bar{s}$ and $\hat{s}$. The initial state is deterministically s_1, *i.e.*, $\beta(s_1) = 1$.

Now, we define the set of actions available in each node. In a state s_i, $i \in N$, there are two actions available a_T and a_F that lead deterministically to state $s_{i,T}$ and $s_{i,F}$, respectively. Intuitively, action a_T is related to set the variable i to true, while a_F to set the variable to false. In both state $s_{i,T}$ and $s_{i,F}$ there is only one action $\bar{a}$ available that leads deterministically to state s_{i+1} if $i < n$ and to $\hat{s}$ if $i = n$. In state $\hat{s}$ there is a single action $\hat{a}$ available that leads to $\bar{s}_1$.

States $\bar{s}_i$ follow a similar structure as states s_i. In particular, in state $\bar{s}_i$ there are two actions a_T and a_F available that lead deterministically to state $\bar{s}_{i,T}$ and $\bar{s}_{i,F}$, respectively. In state $\bar{s}_{i,T}$ and state $\bar{s}_{i,F}$ the only action available is $\bar{a}$ that leads deterministically to state $\bar{s}_{i+1}$ if $i < n$ and to $\bar{s}$ if $i = n$. Finally, there are two actions available in $\bar{s}$: $a^\star$ and $a_\emptyset$. Notice that the built MDP is loop-free and always ends after the agent picks an action in state $\bar{s}$. Hence, $H = 4n + 2$

Finally, we are left to set the rewards and costs. The only action with positive reward is $a^\star$, which has reward 1, *i.e*, $r_h(\bar{s}, a^\star) = 1$ (for each $h \in \mathcal{H}$). All the other rewards are set to 0.

With slight abuse of notation, we remove the dependency from h in the cost function. Indeed, this is not required as any state appears only at a specific step. We add two costs constraints for any variable $i \in N$. We denote the two cost function with $c_{i,1}$ and $c_{i,2}$. The cost $c_{i,1}$ is such that $c_{i,1}(s_{i,T}, \bar{a}) = 1$, $c_{i,1}(\bar{s}_{i,T}, \bar{a}) = -1$, and has value zero for any other couple state-action. Similarly, $c_{i,2}$ is such that $c_{i,1}(s_{i,F}, \bar{a}) = 1$, $c_{i,1}(\bar{s}_{i,F}, \bar{a}) = -1$, and has value zero for any other couple state-action.

Finally, we add an additional cost constraint c_q for any clause $q \in Q$ such that:

- $c_q(\hat{s}, \hat{a}) = 1$
- $c_q(\bar{s}, a_\varnothing) = -1$

- $c_q(\bar{s}_i, a_T) = -1$ if "i is true" is one of the literals of the clause q.
- $c_q(\bar{s}_i, a_F) = -1$ if "i is false" is one of the literals of the clause q.
- All the unspecified costs are set to 0.

If. Assume that there exists an assignment that satisfies all the clauses. Then, we show that there exists a safe policy with expected reward 1. We build this policy as follows. In each state s_i and $\bar{s}_i$ take deterministically action a_T if the variable is true and a_F if the variable is false. Moreover, the policy plays deterministically $\hat{a}$ in state $\hat{s}$ and $a^\star$ in state $\bar{s}$. Finally, the policy plays the only available action $\bar{a}$ in all the other states.

Notice that the since both the policy and the transition function are deterministic, the policy induces deterministically a single trajectory. Since this trajectory includes $a^\star$, the expected reward is 1. Hence, we only need to show that the policy is also safe.

For each variable $i \in N$, the constraints $c_{i,1}$ and $c_{i,2}$ are satisfied. Consider constraint $c_{i,1}$. If the policy suffer cost 1 in state $s_{i,T}$, then the policy also ends in state $\bar{s}_{i,T}$ with cost -1. A similar argument holds for constraint $c_{i,2}$.

Moreover, for each clause q, the policy suffers cost 1 in state $\hat{s}$ playing action $\hat{a}$. However, since each clause is satisfied, it must be the case that the trajectory includes a_T in state $\bar{s}_i, a_T$ where "i is true" belongs to the clause, or a_F in state $\bar{s}_i, a_F$ where "i is false' belongs to the clause. Hence, this guarantees that there is at least one couple action-state for which the cost is -1. This clearly guarantees that the cost starting from every node is at most 0 and hence the policy is feasible.

Only If. Assume that there does not exist an assignment that satisfies all the clauses. We prove that all ϵ-safe policies get a reward of at most $2\epsilon n$.

Consider a policy π that is ϵ-safe. We notice that for each trajectory $\tilde{\tau}$ that ends in $s_{i,T}$, it must be the case that the conditional probability of reaching state $\bar{s}_{i,T}$ is at least $1 - \epsilon$. Formally, it holds:

$$\begin{aligned} C_i^\pi(\tilde{\tau}, \bar{a}) &\coloneqq c_i(s_{i,T}, \bar{a}) + p(s_{i+1}|s_{i,T}, \bar{a}) C_i^\pi(\tilde{\tau} \oplus (\bar{a}, s_{i+1})) \\ &= 1 + p(s_{i+1}|s_{i,T}, \bar{a}) C_i^\pi(\tilde{\tau} \oplus (\bar{a}, s_{i+1})) \leq \epsilon, \end{aligned}$$

This clearly implies that for each variable $i \in N$ the probability of playing an action a in state $\bar{s}_i$ conditioned on having played a in state s_i is at least $1 - \epsilon$. By union bound, this implies that with probability at least $1 - \epsilon n$ the action played in $\bar{s}_i$ is the same action played in s_i for each $i \in N$.

Now consider all the partial trajectories $\mathcal{T}^*$ that terminate with $\hat{s}$. For each sequence $\tau \in \mathcal{T}^*$, let D_τ be the probability that, under the policy π, the trajectory repeats the same sequence of actions as the one played before $\hat{s}$. Formally, if actions $\{a_{s_i} \in \{a_T, a_F\}\}_{i \in [N]}$ were played in state $\{s_i\}_i$, then D_τ is the probability that π plays again actions $\{a_{s_i} \in \{a_T, a_F\}\}_{i \in [N]}$ in states $\{\bar{s}_i\}_{i \in [N]}$.

Now, let $\Pr(\tau)$ be the probability of realizing the partial trajectory τ under policy π. It must hold that:

$$\sum_{\tau \in \mathcal{T}^*} \Pr(\tau) D_\tau \geq 1 - \epsilon n$$

For each trajectory τ that terminates in $\hat{s}$, consider the assignment generated by the actions. Formally, if action a_T is played in state s_i then variable i is set to true, otherwise the variable is set to false. Consider a false clause for that assignment (that exists by assumption). Then, it must be the case that the constraint relative to that cost for the partial trajectory τ suffers cost of 1 from playing $\hat{a}$ in $\hat{s}$, and at most $-(1 - D_\tau)$ in expectation from playing action a_T and a_F.

This clearly implies that $a_\varnothing$ is played with probability at least $D_\tau - \epsilon$ conditioned on the partial trajectory τ. Let $\Pr_{a_\varnothing}(\tau)$ be such probability.

Then,

$$\sum_{\tau \in \mathcal{T}^*} \Pr(\tau) \Pr_{a_\varnothing}(\tau) \geq \sum_{\tau \in \mathcal{T}^*} \Pr(\tau)(D_\tau - \epsilon) \geq \sum_{\tau \in \mathcal{T}^*} \Pr(\tau) D_\tau - \epsilon \geq 1 - 2n\epsilon.$$

This implies that the average reward is at most $2\epsilon n$, since the only action with positive reward 1 is opposed to $a_\varnothing$, concluding the proof. □

B Proofs Omitted From Section 5

Lemma 5.1. *Given a CMDP $\mathcal{M}$, the history-dependent policy implemented by Algorithm 1 is safe.*

Proof. Let $\pi : \mathcal{T} \to \Delta(\mathcal{A})$ be the history-dependent policy implemented by Algorithm 1.

In order to prove the result, we first show that the expected costs $\boldsymbol{c}^\tau := \{c_i^\tau\}_{i\in[m]}$ computed by Algorithm 1 are equal to the values defined by the constraint functions $C^\pi_{h,i}$. In particular, we show that, for every $h \in \mathcal{H}$ and $\tau \in \mathcal{T}_h$ of length h, Algorithm 1 ensures that $C^\pi_{h,i}(\tau) = c_i^\tau$ for all $i \in [m]$.

We prove the result above by an inductive argument. As a base case, consider the last step H, and let $\tau = (s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H) \in \mathcal{T}_H$ be any trajectory of length H. By definition of $\mathcal{G}_{H,s_H}$, we have that any $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w}) \in \mathcal{G}_{H,s_H}$ is such that $\boldsymbol{w} = \boldsymbol{0}$ and, thus, $c_i = \sum_{a\in\mathcal{A}} g_a c_{i,H}(s_H, a)$ for every constraint $i \in [m]$. Moreover, the policy π implemented by Algorithm 1 is such that $\pi(a|\tau) = g_a$ for every $a \in \mathcal{A}$. Therefore, since $\boldsymbol{c}^\tau = \boldsymbol{c}$ for some $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w}) \in \mathcal{G}_{H,s_H}$ by construction of Algorithm 1, we have that $C^\pi_{H,i}(\tau) = c_i^\tau$ for every $i \in [m]$.

By induction, let us now assume that the result above holds for some step $h+1$ with $h \in \mathcal{H} \setminus \{H\}$, namely, $C^\pi_{h+1,i}(\tau) = c_i^\tau$ for every constraint $i \in [m]$ and trajectory $\tau \in \mathcal{T}_{h+1}$ of length $h+1$. Next, we prove that this result also holds for step h, for any trajectory $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h. Since the policy π implemented by Algorithm 1 is such that $\pi(a|\tau) = g_a$ for all $a \in \mathcal{A}$, for every $i \in [m]$ it holds:

$$
\begin{aligned}
C^\pi_{h,i}(\tau) &= \sum_{a\in\mathcal{A}} \pi(a|\tau) \left[c_{h,i}(s_h, a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h, a) C^\pi_{h+1,i}(\tau \oplus (a, s')) \right] \\
&= \sum_{a\in\mathcal{A}} g_a \left[c_{h,i}(s_h, a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h, a) C^\pi_{h+1,i}(\tau \oplus (a, s')) \right] \\
&= \sum_{a\in\mathcal{A}} g_a \left[c_{h,i}(s_h, a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h, a) c_i^{\tau\oplus(a,s')} \right],
\end{aligned}
$$

where the last equality holds by induction. Moreover, by definition of $\mathcal{G}_{h,s_h}$, we have that:

$$
c_i^\tau = \sum_{a\in\mathcal{A}} g_a \left[c_{i,h}(s_h, a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h, a) w_{i,a,s'} \right],
$$

which is equal to $C^\pi_{h,i}(\tau)$ since $\boldsymbol{w}_{a,s'} = \boldsymbol{c}^{\tau\oplus(a,s')}$ by construction.

In conclusion, by Constraints (1d) in the definition of the sets $\mathcal{G}_{h,s}$, we immediately get that:

$$
C^\pi_{h,i}(\tau) \leq 0 \quad \forall i \in [m], \forall h \in \mathcal{H}, \forall \tau \in \mathcal{T}_h,
$$

which concludes the proof. □

Lemma 5.2. *Given a CMDP $\mathcal{M}$, the history-dependent policy $\pi : \mathcal{T} \to \Delta(\mathcal{A})$ implemented by Algorithm 1 is such that $V^\pi = \text{OPT}$.*

Proof. By an inductive argument resembling the one used in the proof of Lemma 5.1, it is possible to show that, for every step $h \in \mathcal{H}$ and trajectory $\tau \in \mathcal{T}_h$ of length h, it holds $V^\pi_h(\tau) = v^\tau$.

For any initial state $s_1 \in \mathcal{S}$, since $v^{(s_1)}$ is obtained by Algorithm 1 by maximizing the value of v over all the pairs $(v, \boldsymbol{c}) \in \mathcal{G}_{1,s_1}$, we have that $v^{(s_1)} = \max_\pi V^\pi_1((s_1))$. Thus, since s_1 is sampled according to β, we can conclude that $V^\pi = \text{OPT}$. □

C Proofs Omitted From Section 6

We first establish the property of the dynamic programming values that is needed in the proof of Lemma 6.1.

Lemma C.1 (Concavity and upward rounding)**.** *Fix $h \in \mathcal{H}$ and $s \in \mathcal{S}$, and keep the set-valued function $\mathcal{F}_{h+1}$ computed by Algorithm 2 fixed. For any $\boldsymbol{x} \in \mathbb{R}^m$, let $\varphi_{h,s}(\boldsymbol{x})$ be the optimal value of $\mathcal{P}^{\delta}_{h,s}(\boldsymbol{x}, \mathcal{F}_{h+1})$ whenever this problem is feasible, and let $B_{h,s}$ be its feasible domain. Then $B_{h,s}$ is convex and upward-closed under the componentwise order, and $\varphi_{h,s}$ is concave and componentwise nondecreasing on $B_{h,s}$. Moreover, for every $(z, \boldsymbol{w}) \in \mathcal{F}_h(s)$, the vector $\lceil \boldsymbol{w} \rceil_\delta$ belongs to $\mathcal{D}^m_\delta \cap B_{h,s}$, and*

$$z \le \varphi_{h,s}(\boldsymbol{w}) \le \varphi_{h,s}(\lceil \boldsymbol{w} \rceil_\delta) = M_h(s, \lceil \boldsymbol{w} \rceil_\delta).$$

In particular, the table entries $M_h(s, \lceil \boldsymbol{w} \rceil_\delta)$ and $G_h(s, \lceil \boldsymbol{w} \rceil_\delta)$ are defined.

Proof. If any $\mathcal{F}_{h+1}(s')$ is empty, then $B_{h,s}$ and $\mathcal{F}_h(s)$ are empty, and the claims are vacuous. Otherwise, write $\mathcal{F}_{h+1}(s') = \mathrm{co}\{(z^{s',j}, \boldsymbol{w}^{s',j}) : j \in J_{s'}\}$, where $J_{s'}$ is a finite nonempty index set. At $h = H$, this set consists of the single point $(0, \mathbf{0})$. Introduce nonnegative variables $t_{a,s',j}$ representing the product of g_a and the weight of point j in the continuation chosen for (a, s'). Then $\mathcal{P}^{\delta}_{h,s}(\boldsymbol{x}, \mathcal{F}_{h+1})$ has the same optimal value and feasibility status as the following linear program:

$$\begin{aligned}
\max_{\boldsymbol{g},t} \quad & \sum_{a\in\mathcal{A}} g_a r_h(s,a) + \sum_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} p_h(s'|s,a) \sum_{j\in J_{s'}} t_{a,s',j} z^{s',j} \\
\text{s.t.} \quad & \sum_{a\in\mathcal{A}} g_a c_{i,h}(s,a) + \sum_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} p_h(s'|s,a) \sum_{j\in J_{s'}} t_{a,s',j} w_i^{s',j} \le x_i + \delta \qquad \forall i \in [m], \\
& \sum_{j\in J_{s'}} t_{a,s',j} = g_a \qquad\qquad \forall a \in \mathcal{A},\ s' \in \mathcal{S}, \\
& \sum_{a\in\mathcal{A}} g_a = 1, \qquad g_a \ge 0, \qquad t_{a,s',j} \ge 0.
\end{aligned}$$

Indeed, a feasible solution to the original problem gives a feasible solution to this LP by multiplying its convex-combination weights by g_a. Conversely, when $g_a > 0$, dividing $t_{a,s',j}$ by g_a recovers a point of $\mathcal{F}_{h+1}(s')$. When $g_a = 0$, all corresponding $t_{a,s',j}$ vanish, and we may choose any point of the nonempty polytope $\mathcal{F}_{h+1}(s')$. These transformations preserve the objective and expected costs.

All LP constraints are linear, and $\boldsymbol{x}$ appears only on the right-hand side of the cost bounds. Thus, a convex combination of feasible solutions for $\boldsymbol{x}$ and $\boldsymbol{y}$ is feasible, proving that $B_{h,s}$ is convex. The LP feasible set is compact whenever nonempty, so optimal solutions exist. Combining optimal solutions gives, for every $\alpha \in [0,1]$,

$$\varphi_{h,s}(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \ge \alpha\varphi_{h,s}(\boldsymbol{x}) + (1-\alpha)\varphi_{h,s}(\boldsymbol{y}),$$

which proves concavity. Increasing a component only relaxes its cost bound, proving upward closure and componentwise monotonicity.

Finally, by construction of $\mathcal{F}_h(s)$, every $(z, \boldsymbol{w}) \in \mathcal{F}_h(s)$ can be written as

$$(z, \boldsymbol{w}) = \sum_j \lambda_j \big(M_h(s, \tilde{\boldsymbol{c}}^j), \tilde{\boldsymbol{c}}^j\big), \qquad \lambda_j \ge 0, \quad \sum_j \lambda_j = 1,$$

where each $\tilde{\boldsymbol{c}}^j \in \mathcal{D}^m_\delta \cap B_{h,s}$ and $M_h(s, \tilde{\boldsymbol{c}}^j) = \varphi_{h,s}(\tilde{\boldsymbol{c}}^j)$. Convexity and upward closure imply that both $\boldsymbol{w}$ and $\lceil \boldsymbol{w} \rceil_\delta$ belong to $B_{h,s}$. Since $\boldsymbol{w}$ lies in the convex hull of grid vectors, rounding it upward stays within $\mathcal{D}^m_\delta$. Jensen's inequality and monotonicity now yield

$$z = \sum_j \lambda_j \varphi_{h,s}(\tilde{\boldsymbol{c}}^j) \le \varphi_{h,s}\left(\sum_j \lambda_j \tilde{\boldsymbol{c}}^j\right) = \varphi_{h,s}(\boldsymbol{w}) \le \varphi_{h,s}(\lceil \boldsymbol{w} \rceil_\delta).$$

At this feasible grid vector, Algorithm 2 stores the optimal value and an optimal action distribution, proving the remaining claims. □

Lemma 6.1. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$, let $\pi : \mathcal{T} \times \mathcal{A} \to [0,1]$ be the history-dependent policy implemented by Algorithm 3. Then, for every step $h \in \mathcal{H}$ and trajectory $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h, it holds that $V^\pi_h(\tau) \ge M_h(s_h, \tilde{\boldsymbol{c}}^\tau)$, where $\tilde{\boldsymbol{c}}^\tau$ is defined in Algorithm 3.*

Proof. First, the table lookups in Algorithm 3 are well-defined. Lemma 6.2, whose proof does not use the present lemma, gives feasibility at the rounded-down continuation costs of a safe optimal policy. These costs are componentwise nonpositive, so upward closure in Lemma C.1 gives feasibility at the initial budget $\mathbf{0}$. At each subsequent step, the same lemma ensures that the rounded continuation budget stored in W_h indexes defined entries of M_{h+1} and G_{h+1}.

We prove the result by induction.

As a base case for the induction, let us consider the last step H. In such a case, for every trajectory $\tau = (s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H) \in \mathcal{T}_H$ of length H, Algorithm 3 ensures that:

$$\begin{aligned} V_H^\pi(\tau) &= \sum_{a \in \mathcal{A}} \pi(a|\tau) r_H(s_H, a) \\ &= \sum_{a \in \mathcal{A}} g_a r_H(s_H, a) && \text{(with } \boldsymbol{g} = G_H(s_H, \tilde{\boldsymbol{c}}^\tau)\text{)} \\ &= M_H(s_H, \tilde{\boldsymbol{c}}^\tau), \end{aligned}$$

where the first equality holds by definition of $V_H^\pi(\tau)$, the second one by the way in which Algorithm 3 selects the action a_H to be played at step H, and the third one by how $M_H(s_H, \tilde{\boldsymbol{c}}^\tau)$ and $G_H(s_H, \tilde{\boldsymbol{c}}^\tau)$ are computed by Algorithm 2, during the iteration in which $\tilde{\boldsymbol{c}} = \tilde{\boldsymbol{c}}^\tau$. Specifically, $M_H(s_H, \tilde{\boldsymbol{c}}^\tau)$ and $G_H(s_H, \tilde{\boldsymbol{c}}^\tau)$ are obtained by extracting v and $\boldsymbol{g}$, respectively, from a solution $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ to problem $\mathcal{P}^\delta_{H,s_H}(\tilde{\boldsymbol{c}}^\tau, \mathcal{F}_{H+1})$. Thus, by noticing that the polytopes $\mathcal{F}_{H+1}(s)$ defined by Algorithm 2 are such that $\mathcal{F}_{H+1}(s) = (0, \mathbf{0})$ for all $s \in \mathcal{S}$, the third equality above is trivially satisfied.

Now, consider any step $h \in \mathcal{H} \setminus \{H\}$ and assume by induction that the result holds for step $h+1$, for all the trajectories of length $h+1$. Then, for every trajectory $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h, we have that:

$$V_h^\pi(\tau) = \sum_{a \in \mathcal{A}} \pi(a|\tau) \left[r_h(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) V_{h+1}^\pi(\tau \oplus (a, s')) \right] \tag{3a}$$

$$\geq \sum_{a \in \mathcal{A}} \pi(a|\tau) \left[r_h(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) M_{h+1}(s', \tilde{\boldsymbol{c}}^{\tau \oplus (a,s')}) \right] \tag{3b}$$

$$= \sum_{a \in \mathcal{A}} g_a \left[r_h(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) M_{h+1}(s', \tilde{\boldsymbol{c}}^{\tau \oplus (a,s')}) \right] \text{ (with } \boldsymbol{g} = G_h(s_h, \tilde{\boldsymbol{c}}^\tau)\text{)} \tag{3c}$$

$$= \sum_{a \in \mathcal{A}} g_a \left[r_h(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) M_{h+1}(s', W_h(s_h, \tilde{\boldsymbol{c}}^\tau, a, s')) \right] \tag{3d}$$

$$= \sum_{a \in \mathcal{A}} g_a \left[r_h(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) M_{h+1}(s', \lceil \boldsymbol{w}_{a,s'} \rceil_\delta) \right] \tag{3e}$$

$$\geq \sum_{a \in \mathcal{A}} g_a \left[r_h(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) z_{a,s'} \right] \tag{3f}$$

$$= M_h(s_h, \tilde{\boldsymbol{c}}^\tau). \tag{3g}$$

Equation (3a) holds by definition of $V_h^\pi(\tau)$. Equation (3b) holds by induction, namely, it follows from the fact that $V_{h+1}^\pi(\tau \oplus (a, s')) \geq M_{h+1}(s', \tilde{\boldsymbol{c}}^{\tau \oplus (a,s')})$ for all $a \in \mathcal{A}$ and $s' \in \mathcal{S}$. Equation (3c) holds by the way in which Algorithm 3 selects the action a_h to be played at step h, while Equation (3d) by definition of $\tilde{\boldsymbol{c}}^{\tau \oplus (a,s')}$ (see Algorithm 3). Equation (3e) holds by how $W_h(s_h, \tilde{\boldsymbol{c}}^\tau, a, s')$ is computed by Algorithm 2, with $\boldsymbol{w}_{a,s'}$ being obtained by the algorithm from an optimal solution $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ to problem $\mathcal{P}^\delta_h(\tilde{\boldsymbol{c}}^\tau, \mathcal{F}_{h+1})$. Equation (3f) follows from Lemma C.1, applied at step $h+1$ and state s'. Indeed, feasibility of the solution computed by Algorithm 2 gives $(z_{a,s'}, \boldsymbol{w}_{a,s'}) \in \mathcal{F}_{h+1}(s')$ for every $a \in \mathcal{A}$ and $s' \in \mathcal{S}$. Hence,

$$z_{a,s'} \leq \varphi_{h+1,s'}(\boldsymbol{w}_{a,s'}) \leq M_{h+1}(s', \lceil \boldsymbol{w}_{a,s'} \rceil_\delta),$$

where the rounded budget indexes a defined table entry by the same lemma. Multiplying by the nonnegative weights $g_a p_h(s'|s_h, a)$ and summing proves the inequality. Finally, Equation (3g) holds by the way in which $M_h(s_h, \tilde{\boldsymbol{c}}^\tau)$ is computed by Algorithm 2. □

Lemma 6.2. *Let $\pi^\star : \mathcal{T} \times \mathcal{A} \to [0,1]$ be an optimal history-dependent policy in a CMDP $\mathcal{M}$ subject to trajectory safety requirements. Then, given $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, Algorithm 3 ensures that $M_h(s_h, \lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta) \geq V_h^{\pi^\star}(\tau)$ for every step $h \in \mathcal{H}$ and trajectory $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h, where δ is defined in Algorithm 3 and we let $\boldsymbol{c}^{\star,\tau} \coloneqq \{c_i^{\star,\tau}\}$ with $c_i^{\star,\tau} \coloneqq C_{h,i}^{\pi^\star}(\tau)$.*

Proof. We prove the result by induction.

For the ease of presentation, for every $h \in \mathcal{H}$, $s \in \mathcal{S}$, and $\tilde{\boldsymbol{c}} \in \bigtimes_{i \in [m]} \mathcal{D}_\delta$, in the following we let $\Psi_{h,s}^\delta(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1}) \subseteq \mathbb{R}^{1+m+|\mathcal{A}|+(1+m)\cdot|\mathcal{A}|\cdot|\mathcal{S}|}$ be the set containing all the tuples $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ that satisfy Constraints (2b)–(2g) in problem $\mathcal{P}_{h,s}^\delta(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$. Intuitively, this set encompasses all the tuples that are feasible for the problem, and, thus, it represents its feasibility region.

As a base case, consider the last step H and any trajectory $\tau = (s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H) \in \mathcal{T}_H$ of length H. Let $\boldsymbol{g}^\star \coloneqq \{g_a^\star\}_{a \in \mathcal{A}}$ with $g_a^\star \coloneqq \pi^\star(a|\tau)$, $\boldsymbol{z}^\star \coloneqq \boldsymbol{0}$, and $\boldsymbol{w}^\star \coloneqq \boldsymbol{0}$. Moreover, let $v^\star$ and $\boldsymbol{c}^\star \coloneqq \{c_i^\star\}_{i \in [m]}$ be obtained by plugging $\boldsymbol{g} = \boldsymbol{g}^\star$, $\boldsymbol{z} = \boldsymbol{z}^\star$, and $\boldsymbol{w} = \boldsymbol{w}^\star$ into the right-hand sides of the equations in Constraints (2b) and (2c). It is easy to see that $(v^\star, \boldsymbol{c}^\star, \boldsymbol{g}^\star, \boldsymbol{z}^\star, \boldsymbol{w}^\star) \in \Psi_{H,s_H}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{H+1})$. Indeed, Constraints (2e) are clearly satisfied since, for every $i \in [m]$:

$$c_i^\star = \sum_{a \in \mathcal{A}} g_a^\star c_{H,i}(s_H, a) = C_{H,i}^{\pi^\star}(\tau) = c_i^{\star,\tau} \leq \lfloor c_i^{\star,\tau} \rfloor_\delta + \delta.$$

Let $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ be the optimal solution to problem $\mathcal{P}_{H,s_H}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{H+1})$ computed in Line 9 of Algorithm 2 when $\tilde{\boldsymbol{c}} = \lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta$. Then, the following holds:

$$\begin{aligned}
M_H(s_H, \lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta) = v &= \sum_{a \in \mathcal{A}} g_a r_H(s_H, a) \\
&\geq v^\star = \sum_{a \in \mathcal{A}} g_a^\star r_H(s_H, a) \text{ (since } (v^\star, \boldsymbol{c}^\star, \boldsymbol{g}^\star, \boldsymbol{z}^\star, \boldsymbol{w}^\star) \in \Psi_{H,s_H}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{H+1})) \\
&= \sum_{a \in \mathcal{A}} \pi^\star(a|\tau) r_H(s_H, a) \\
&= V_H^{\pi^\star}(\tau),
\end{aligned}$$

where the inequality holds since $(v^\star, \boldsymbol{c}^\star, \boldsymbol{g}^\star, \boldsymbol{z}^\star, \boldsymbol{w}^\star) \in \Psi_{H,s_H}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{H+1})$, *i.e.*, it is feasible for problem $\mathcal{P}_{H,s_H}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{H+1})$, and $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ is optimal for problem $\mathcal{P}_{H,s_H}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{H+1})$.

As for the inductive step, let us consider any step $h \in \mathcal{H} \setminus \{H\}$ and assume that the result holds for the following step $h+1$, for all the trajectories of length $h+1$. Take any trajectory $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h. Let $\boldsymbol{g}^\star \coloneqq \{g_a^\star\}_{a \in \mathcal{A}}$ with $g_a^\star \coloneqq \pi^\star(a|\tau)$. Moreover, let $\boldsymbol{w}^\star \coloneqq \{w_{i,a,s'}^\star\}_{i \in [m], a \in \mathcal{A}, s' \in \mathcal{S}}$ and $\boldsymbol{z}^\star \coloneqq \{z_{a,s'}^\star\}_{a \in \mathcal{A}, s' \in \mathcal{S}}$ be such that:

$$w_{i,a,s'}^\star \coloneqq \lfloor c_i^{\star,\tau \oplus (a,s')} \rfloor_\delta \quad \text{and} \quad z_{a,s'}^\star \coloneqq M_{h+1}(s', \lfloor \boldsymbol{c}^{\star,\tau \oplus (a,s')} \rfloor_\delta).$$

Finally, let $v^\star$ and $\boldsymbol{c}^\star \coloneqq \{c_i^\star\}_{i \in [m]}$ be obtained by plugging $\boldsymbol{g} = \boldsymbol{g}^\star$, $\boldsymbol{z} = \boldsymbol{z}^\star$, and $\boldsymbol{w} = \boldsymbol{w}^\star$ into the right-hand sides of the equations in Constraints (2b) and (2c).

As a first step, we prove that $(v^\star, \boldsymbol{c}^\star, \boldsymbol{g}^\star, \boldsymbol{z}^\star, \boldsymbol{w}^\star) \in \Psi_{h,s_h}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{h+1})$. First, notice that Constraints (2b)–(2b) and Constraints (2f)–(2g) are satisfied by construction. Moreover, Constraints (2d) are satisfied since, for every $a \in \mathcal{A}$ and $s' \in \mathcal{S}$, it holds that $\boldsymbol{w}_{a,s'}^\star \coloneqq \{w_{i,a,s'}^\star\}_{i \in [m]}$ belongs to the set $\mathcal{D}_\delta^m$ and $z_{a,s'}^\star = M_{h+1}(s', \boldsymbol{w}_{a,s'}^\star)$, and, thus, $(z_{a,s'}^\star, \boldsymbol{w}_{a,s'}^\star) \in \mathcal{F}_{h+1}(s')$ by the way in which the polytope $\mathcal{F}_{h+1}(s')$ is constructed for step $h+1$ by Algorithm 2. It remains to show that Constraints (2e) are satisfied. Indeed, by letting $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ be the optimal solution to problem $\mathcal{P}_{h,s_h}^\delta(\lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta, \mathcal{F}_{h+1})$ computed in Line 9 of Algorithm 2 when $\tilde{\boldsymbol{c}} = \lfloor \boldsymbol{c}^{\star,\tau} \rfloor_\delta$, we have that:

$$c_i^\star = \sum_{a \in \mathcal{A}} g_a^\star \left[c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) w_{i,a,s'}^\star \right]$$

$$
\begin{aligned}
&= \sum_{a\in\mathcal{A}} \pi^\star(a|\tau)\left[c_{h,i}(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a)\lfloor c_i^{\star,\tau\oplus(a,s')}\rfloor_\delta\right] && \text{(by def. of } \boldsymbol{g}^\star \text{ and } \boldsymbol{w}^\star)\\
&\leq \sum_{a\in\mathcal{A}} \pi^\star(a|\tau)\left[c_{h,i}(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a) c_i^{\star,\tau\oplus(a,s')}\right] && \text{(by def. of } \lfloor\cdot\rfloor_\delta)\\
&= \sum_{a\in\mathcal{A}} \pi^\star(a|\tau)\left[c_{h,i}(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a) C_{h+1,i}^{\pi^\star}(\tau\oplus(a,s'))\right] && \text{(by def. of } c_i^{\star,\tau\oplus(a,s')})\\
&= C_{h,i}^{\pi^\star}(\tau) && \text{(by def. of } C_{h,i}^{\pi^\star}(\tau))\\
&= c_i^{\star,\tau} && \text{(by def. of } c_i^{\star,\tau})\\
&\leq \lfloor c_i^{\star,\tau}\rfloor_\delta + \delta.
\end{aligned}
$$

This shows that $(v^\star, \boldsymbol{c}^\star, \boldsymbol{g}^\star, \boldsymbol{z}^\star, \boldsymbol{w}^\star) \in \Psi^\delta_{h,s_h}(\lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta, \mathcal{F}_{h+1})$.

Now, we show that $M_h(s_h, \lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta) \geq V_h^{\pi^\star}(\tau)$. By letting $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ be the optimal solution to problem $\mathcal{P}^\delta_{h,s_h}(\lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta, \mathcal{F}_{h+1})$ computed in Line 9 of Algorithm 2 when $\tilde{\boldsymbol{c}} = \lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta$, we have that:

$$
\begin{aligned}
M_h(s_h, \lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta) = v &= \sum_{a\in\mathcal{A}} g_a\left[r_h(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a) z_{a,s'}\right]\\
&\geq v^\star = \sum_{a\in\mathcal{A}} g_a^\star\left[r_h(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a) z_{a,s'}^\star\right] && (\star)\\
&= \sum_{a\in\mathcal{A}} g_a^\star\left[r_h(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a) M_{h+1}(s', \lfloor \boldsymbol{c}^{\star,\tau\oplus(a,s')}\rfloor_\delta)\right] && \text{(def. of } \boldsymbol{z}^\star)\\
&\geq \sum_{a\in\mathcal{A}} g_a^\star\left[r_h(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a) V_{h+1}^{\pi^\star}(\tau\oplus(a,s'))\right] && \text{(induction)}\\
&= \sum_{a\in\mathcal{A}} \pi^\star(a|\tau)\left[r_h(s_h,a) + \sum_{s'\in\mathcal{S}} p_h(s'|s_h,a) V_{h+1}^{\pi^\star}(\tau\oplus(a,s'))\right] && \text{(def. of } \boldsymbol{g}^\star)\\
&= V_h^{\pi^\star}(\tau),
\end{aligned}
$$

where step $(\star)$ holds since $(v^\star, \boldsymbol{c}^\star, \boldsymbol{g}^\star, \boldsymbol{z}^\star, \boldsymbol{w}^\star) \in \Psi^\delta_{h,s_h}(\lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta, \mathcal{F}_{h+1})$, *i.e.*, it is feasible for problem $\mathcal{P}^\delta_{h,s_h}(\lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta, \mathcal{F}_{h+1})$, and $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ is optimal for problem $\mathcal{P}^\delta_{h,s_h}(\lfloor \boldsymbol{c}^{\star,\tau}\rfloor_\delta, \mathcal{F}_{h+1})$.

This concludes the proof. □

Lemma 6.3. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, the history-dependent policy implemented by Algorithm 3 is trajectory ϵ-safe.*

Proof. Let $\pi : \mathcal{T}\times\mathcal{A} \to [0,1]$ be the history-dependent policy implemented by Algorithm 3.

We prove that π is ϵ-safe by induction. In particular, we show that, for every step $h \in \mathcal{H}$ and trajectory $\tau \in \mathcal{T}_h$ of length h, it holds $C_{h,i}^\pi(\tau) \leq \tilde{c}_i^\tau + 2\delta(H-h+1)$ for all $i \in [m]$, where we let $\tilde{c}_i^\tau$ be the i-th element of the vector $\tilde{\boldsymbol{c}}^\tau \coloneqq \{\tilde{c}_i^\tau\}_{i\in[m]}$ of discretized expected costs defined by Algorithm 3.

As a base case for the induction, let us consider the last step H. Then, for every trajectory $\tau = (s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H) \in \mathcal{T}_H$ of length H, the following holds for every $i \in [m]$:

$$
\begin{aligned}
C_{H,i}^\pi(\tau) &= \sum_{a\in\mathcal{A}} \pi(a|\tau) c_{H,i}(s_H,a) && \text{(by definiton of } C_{H,i}^\pi(\tau))\\
&= \sum_{a\in\mathcal{A}} g_a c_{H,i}(s_H,a) && \text{(with } \boldsymbol{g} = G_H(s_H, \tilde{\boldsymbol{c}}^\tau))\\
&\leq \tilde{c}_i^\tau + \delta\\
&\leq \tilde{c}_i^\tau + 2\delta,
\end{aligned}
$$

where the second equality above holds by the way in which Algorithm 3 selects the action a_H to be played at step H, while the inequality holds since $\boldsymbol{g}$ is obtained by Algorithm 2 from an optimal solution $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ to problem $\mathcal{P}^{\delta}_{H,s_H}(\tilde{\boldsymbol{c}}^{\tau}, \mathcal{F}_{H+1})$. Thus, $c_i = \sum_{a \in \mathcal{A}} g_a c_{H,i}(s_H, a) \leq \tilde{c}^{\tau}_i + \delta$ for every $i \in [m]$, by Constraints (2c) and (2e) in problem $\mathcal{P}^{\delta}_{H,s_H}(\tilde{\boldsymbol{c}}^{\tau}, \mathcal{F}_{H+1})$.

Now, let us consider any step $h \in \mathcal{H} \setminus \{H\}$ and assume that the result holds for the following step $h+1$, for any trajectory of length $h+1$. For every trajectory $\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1}, s_h) \in \mathcal{T}_h$ of length h, the following holds for every $i \in [m]$:

$$
\begin{aligned}
C^{\pi}_{h,i}(\tau) &= \sum_{a \in \mathcal{A}} \pi(a|\tau) \left[c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) C^{\pi}_{h+1,i}(\tau \oplus (a, s')) \right] && \text{(4a)} \\
&= \sum_{a \in \mathcal{A}} g_a \left[c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) C^{\pi}_{h+1,i}(\tau \oplus (a, s')) \right] && \text{(4b)} \\
&\leq \sum_{a \in \mathcal{A}} g_a \left[c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) \left(\tilde{c}^{\tau \oplus (a,s')}_i + 2\delta(H-h) \right) \right] && \text{(4c)} \\
&= \sum_{a \in \mathcal{A}} g_a \left[c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) \left([W_h(s_h, \tilde{\boldsymbol{c}}^{\tau}, a, s')]_i + 2\delta(H-h) \right) \right] && \text{(4d)} \\
&= \sum_{a \in \mathcal{A}} g_a \left[c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) \left(\lceil w_{i,a,s'} \rceil_{\delta} + 2\delta(H-h) \right) \right] && \text{(4e)} \\
&\leq \sum_{a \in \mathcal{A}} g_a \left[c_{h,i}(s_h, a) + \sum_{s' \in \mathcal{S}} p_h(s'|s_h, a) \left(w_{i,a,s'} + \delta + 2\delta(H-h) \right) \right] && \text{(4f)} \\
&= c_i + \delta + 2\delta(H-h) && \text{(4g)} \\
&\leq \tilde{c}^{\tau}_i + 2\delta + 2\delta(H-h) && \text{(4h)} \\
&= \tilde{c}^{\tau}_i + 2\delta(H-h+1). && \text{(4i)}
\end{aligned}
$$

Equation (4a) holds by definition of $C^{\pi}_{h,i}(\tau)$. Equation (4b), where $\boldsymbol{g} = G_h(s_h, \tilde{\boldsymbol{c}}^{\tau})$, holds by the way in which Algorithm 3 selects the action a_h to be played at step h. Equation (4c) holds by induction, namely $C^{\pi}_{h+1,i}(\tau \oplus (a, s')) \leq \tilde{c}^{\tau \oplus (a,s')}_i + 2\delta[H - (h+1) + 1]$. Equation (4d) holds by how Algorithm 3 defines $\tilde{\boldsymbol{c}}^{\tau \oplus (a,s')}$, where $[W_h(s_h, \tilde{\boldsymbol{c}}^{\tau}, a, s')]_i$ denotes the i-th component of $W_h(s_h, \tilde{\boldsymbol{c}}^{\tau}, a, s')$, while Equation (4e) by how the values $W_h(s_h, \tilde{\boldsymbol{c}}^{\tau}, a, s')$ for $a \in \mathcal{A}$ and $s' \in \mathcal{S}$ are computed by Algorithm 2. Equation (4f) holds by definition of $\lceil \cdot \rceil_{\delta}$. Finally, Equations (4g) and (4h) hold since $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ is an optimal solution to problem $\mathcal{P}^{\delta}_{h,s_h}(\tilde{\boldsymbol{c}}^{\tau}, \mathcal{F}_{h+1})$, and in particular by Constraints (2c) and (2e).

To conclude the proof, it is sufficient to notice that $\delta \leq \epsilon/2H$ by Algorithm 3, and, thus, it holds $2\delta(H-h+1) \leq \epsilon$ for all $h \in \mathcal{H}$. □

Theorem 6.4. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, the history-dependent policy $\pi : \mathcal{T} \times \mathcal{A} \to [0, 1]$ implemented by Algorithm 3 is trajectory ϵ-safe and achieves $V^{\pi} \geq \text{OPT}$.*

Proof. By Lemma 6.3, the history-dependent policy π implemented by Algorithm 3 is trajectory ϵ-safe. Moreover, by Lemma 6.1 and Lemma 6.2, for every initial state $s_1 \in \mathcal{S}$, by letting $\tau = (s_1)$ it holds:

$$
\begin{aligned}
V^{\pi}_1(\tau) &\geq M_1(s_1, \tilde{\boldsymbol{c}}^{\tau}) && \text{(Lemma 6.1)} \\
&= M_1(s_1, \boldsymbol{0}) && \text{(definition of } \tilde{\boldsymbol{c}}^{\tau} \text{ for } \tau = (s_1)) \\
&\geq M_1(s_1, \lfloor \boldsymbol{c}^{\star,\tau} \rfloor_{\delta}) && \text{(feasibility set of } \mathcal{P}^{\delta}_{1,s_1}(\tilde{\boldsymbol{c}}, \mathcal{F}_1) \text{ shrinks when } \tilde{\boldsymbol{c}} \text{ goes from } \boldsymbol{0} \text{ to } \lfloor \boldsymbol{c}^{\star,\tau} \rfloor_{\delta}) \\
&\geq V^{\pi^{\star}}_1(\tau). && \text{(Lemma 6.2)}
\end{aligned}
$$

Thus, it follows that:

$$
V^{\pi} = \sum_{s \in \mathcal{S}} \beta(s) V^{\pi}_1((s))
$$

$$
\begin{aligned}
&\geq \sum_{s \in \mathcal{S}} \beta(s) V_1^{\pi^\star}((s)) \\
&= V^{\pi^\star} \\
&\geq \text{OPT}.
\end{aligned}
$$

This concludes the proof. □

Theorem 6.5. *Given a CMDP $\mathcal{M}$ and an error $\epsilon > 0$ as inputs, Algorithm 3 runs in time polynomial in $1/\epsilon$ and the size of the CMDP $\mathcal{M}$, when the number of constraints m is constant.*

Proof. Clearly, all the operations performed by Algorithm 3 can be done in $\mathcal{O}(1)$ time. Thus, we only need to prove that Algorithm 2 runs in time polynomial in $1/\epsilon$ and the size of the CMDP $\mathcal{M}$.

The main loop of Algorithm 2 is executed for each triplet $h \in \mathcal{H}$, $s \in \mathcal{S}$, and $\tilde{\boldsymbol{c}} \in \mathcal{D}_\delta^m$. Thus, since Algorithm 2 is given as input a step size $\delta = \min\{\epsilon/2H, 1/2\}$ and the cardinality of the set $\mathcal{D}_\delta^m$ is $(\lceil H/\delta \rceil + 1)^m$, when the number of constraints m is constant, the number of iterations of the loop is polynomial in $1/\epsilon$ and the size of the CMDP $\mathcal{M}$ given as input to the algorithm.

All the operations done in one iteration of the main loop of Algorithm 2 can be done in $\mathcal{O}(1)$ time, except for those at Lines 9 and 16. In particular, at Line 9, Algorithm 2 has to compute an optimal solution to problem $\mathcal{P}_{h,s}^\delta(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$, while, at Line 16, it has to build the H-representation of the convex hull of a suitably-defined set $\mathcal{V}$ of $(v, \tilde{\boldsymbol{c}})$ pairs with $\tilde{\boldsymbol{c}} \in \mathcal{D}_\delta^m$.

The H-representation of the convex hull of a set of points can be easily built in polynomial time by leveraging its extended formulation. In general, given a set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of n points in $\mathbb{R}^d$, the extended representation of the convex hull is the following polytope:

$$
\left\{ (\boldsymbol{u}, \boldsymbol{y}) \in \mathbb{R}^{d+n} \mid \boldsymbol{u} = \sum_{i=1}^n y_i \boldsymbol{x}_i \wedge \sum_{i=1}^n y_i = 1 \wedge y_i \in [0,1] \ \forall i \in [n] \right\}. \tag{5}
$$

It is clear that the H-representation of this polytope is polynomial in d and n, and that its projection over the first component $\boldsymbol{u}$ is the convex hull of the points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. For the specific case of Line 16 in Algorithm 2, the number of points n is at most $(\lceil H/\delta \rceil + 1)^m$, and, thus, the (extended) H-representation of the convex hull of $\mathcal{V}$ can be built in polynomial time when the number of constraints m is constant. Without the extra variables, the H-representation could be exponential in m and n.

The problem $\mathcal{P}_{h,s}^\delta(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ at Line 9 can be solved in polynomial time by solving a suitable relaxed problem, which is an LP. For every $h \in \mathcal{H}$, $s \in \mathcal{S}$, and $\tilde{\boldsymbol{c}} \in \mathcal{D}_\delta^m$, the relaxed version $\mathcal{R}_{h,s}^\delta(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ of problem $\mathcal{P}_{h,s}^\delta(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ reads as follows:

$$
\begin{aligned}
\max_{v, \boldsymbol{c}, \boldsymbol{g}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{w}}} \quad & v \quad \text{s.t.} && && \text{(6a)} \\
& v = \sum_{a \in \mathcal{A}} g_a r_h(s,a) + \sum_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p_h(s'|s,a) \tilde{z}_{a,s'} && && \text{(6b)} \\
& c_i = \sum_{a \in \mathcal{A}} g_a c_{i,h}(s,a) + \sum_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p_h(s'|s,a) \tilde{w}_{i,a,s'} && \forall i \in [m] && \text{(6c)} \\
& (\tilde{z}_{a,s'}, \tilde{\boldsymbol{w}}_{a,s'}) \in \widetilde{\mathcal{F}}_{h+1}(s', g_a) && \forall a \in \mathcal{A}, \forall s' \in \mathcal{S} && \text{(6d)} \\
& c_i \leq \tilde{c}_i + \delta && \forall i \in [m] && \text{(6e)} \\
& \sum_{a \in \mathcal{A}} g_a = 1 && && \text{(6f)} \\
& g_a \geq 0 && \forall a \in \mathcal{A}, && \text{(6g)}
\end{aligned}
$$

which is obtained from $\mathcal{P}_{h,s}^\delta(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ by introducing the variables $\tilde{z}_{a,s'} \coloneqq g_a z_{a,s'}$ and $\tilde{w}_{i,a,s'} \coloneqq g_a w_{i,a,s'}$ for all $i \in [m]$, $a \in \mathcal{A}$, and $s' \in \mathcal{S}$, and by letting:

$$
\widetilde{\mathcal{F}}_{h+1}(s', g_a) \coloneqq \left\{ (\tilde{z}, \tilde{\boldsymbol{w}}, \boldsymbol{y}) \mid \tilde{A}_{h+1,s'} \begin{pmatrix} \tilde{z} \\ \tilde{\boldsymbol{w}} \\ \boldsymbol{y} \end{pmatrix} \leq g_a \tilde{b}_{h+1,s'} \right\},
$$

where the matrix A is defined in Equation (5). It is immediate to see that $\mathcal{R}^{\delta}_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ is a relaxation of $\mathcal{P}^{\delta}_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$.

Notice that we can always discard $\boldsymbol{y}$, which essentially means that we are projecting $\widetilde{\mathcal{F}}_{h+1}$ on its first components. Hence, we can define a (potentially exponentially large [Ziegler, 1995]) matrix $A_{h+1,s'}$ and vector $b_{h+1,s'}$ associated with the first two components and define $\widetilde{\mathcal{F}}_{h+1}$ equivalently as

$$\widetilde{\mathcal{F}}_{h+1}(s', g_a) \coloneqq \left\{ (\tilde{z}, \tilde{\boldsymbol{w}}) \mid A_{h+1,s'} \begin{pmatrix} \tilde{z} \\ \tilde{\boldsymbol{w}} \end{pmatrix} \leq g_a b_{h+1,s'} \right\},$$

Since in the remaining of the proof we do not care about computation time, we use this more compact representation and disregard the extra variables.

Next, we show that, from an optimal solution $(v, \boldsymbol{c}, \boldsymbol{g}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{w}})$ to $\mathcal{R}^{\delta}_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$, it is possible to recover a solution $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ to problem $\mathcal{P}^{\delta}_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$ with at least the same objective value, in polynomial time. Fix $i \in [m]$, $a \in \mathcal{A}$, and $s' \in \mathcal{S}$. Let us first consider the case in which $g_a \neq 0$. Then, we let $z_{a,s'} \coloneqq \tilde{z}_{a,s'}/g_a$ and $w_{i,a,s'} \coloneqq \tilde{w}_{i,a,s'}/g_a$ for all $i \in [m]$ and $s' \in \mathcal{S}$. Instead, in the case in which $g_a = 0$, it is immediate to see that it must be $\tilde{z}_{a,s'} = 0$ and $\tilde{w}_{i,a,s'} = 0$ for all $i \in [m]$ and $s' \in \mathcal{S}$. Indeed, assume by contradiction that $(\tilde{z}_{a,s'}, \tilde{\boldsymbol{w}}_{a,s'}) \neq (0, \boldsymbol{0})$, and write $(\tilde{z}, \tilde{\boldsymbol{w}}) = (\tilde{z}_{a,s'}, \tilde{\boldsymbol{w}}_{a,s'})$. Take any $(z, \boldsymbol{w}) \in \mathcal{F}_{h+1}(s')$ and let $(z, \boldsymbol{w}) + \lambda(\tilde{z}, \tilde{\boldsymbol{w}})$ for any $\lambda \geq 0$, then:

$$\begin{aligned} A_{h+1,s'} \left[\begin{pmatrix} z \\ \boldsymbol{w} \end{pmatrix} + \lambda \begin{pmatrix} \tilde{z} \\ \tilde{\boldsymbol{w}} \end{pmatrix} \right] &= A_{h+1,s'} \begin{pmatrix} z \\ \boldsymbol{w} \end{pmatrix} + \lambda A_{h+1,s'} \begin{pmatrix} \tilde{z} \\ \tilde{\boldsymbol{w}} \end{pmatrix} \\ &\leq b_{h+1,s'} + \lambda g_a b_{h+1,s'} \\ &= b_{h+1,s'}, \end{aligned}$$

which contradicts the fact that $\mathcal{F}_{h+1}(s')$ is a polytope, and, thus, a bounded set. Hence, for every action a with $g_a = 0$ and every $s' \in \mathcal{S}$, we choose any pair $(z_{a,s'}, \boldsymbol{w}_{a,s'}) \in \mathcal{F}_{h+1}(s')$, for example one of its generating points. Such a point exists by Lemma 6.2 and the assumption that a safe policy exists, or by the initialization $\mathcal{F}_{H+1}(s') = \{(0, \boldsymbol{0})\}$ when $h = H$. This choice satisfies the membership constraints and contributes zero to the reward and expected costs because $g_a = 0$. It is immediate to see that the solution $(v, \boldsymbol{c}, \boldsymbol{g}, \boldsymbol{z}, \boldsymbol{w})$ that we have just defined satisfies all the constraints of problem $\mathcal{P}^{\delta}_{h,s}(\tilde{\boldsymbol{c}}, \mathcal{F}_{h+1})$. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state all the main contributions made by the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: All the assumptions are clearly stated.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All the theoretical results clearly state their assumptions, while all their proofs are provided in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [N/A]

Justification: The paper does *not* include experiments.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed, since the work is mainly theoretical.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.