Subprocess-Constrained Markov Decision Processes

Jiarui Gan
University of Oxford
Oxford, UK
jiarui.gan@cs.ox.ac.uk

Debmalya Mandal
University of Warwick
Warwick, UK
debmalya.mandal@warwick.ac.uk

Abstract

Constrained Markov decision processes (CMDPs) typically bound expected cumulative cost from an initial state. Such a guarantee can be insufficient when a policy must remain feasible after every history it may encounter. We introduce *subprocess-constrained MDPs* (S-CMDPs), which require the expected discounted cost of each continuation, conditional on any reachable history, to satisfy a bound determined by the current state. This stronger requirement changes the structure and complexity of the problem: we show that stationary policies can be strictly suboptimal, and finding even an approximately optimal stationary policy is NP-hard. We adapt a recently introduced value-set iteration framework to show that history-dependent policies surprisingly overcome both obstacles. Our polynomial-time algorithm represents such a policy implicitly: its expected reward is within ϵ of the optimal feasible value, it violates each subprocess cost bound by at most δ, and its action distribution can be computed in polynomial time for any given history. Code is available at https://github.com/jrgan/VSI.

1 Introduction

Markov decision processes (MDPs) are a standard model for sequential decision-making under uncertainty. At each step, an agent observes the state of the environment, chooses an action, and receives a reward as the environment evolves stochastically. In many domains, however, decision-making is subject to requirements that are *not* captured by reward alone: energy consumption must stay within a budget, risk must remain below a safety threshold, and violations of service-level targets must be limited. Constrained Markov decision processes (CMDPs) [3] address such requirements by augmenting an MDP with a *cost* function and imposing a constraint, typically on the expected cumulative cost over the entire trajectory. This global, expectation-based formulation has a long history in stochastic control and operations research and remains central to modern safe reinforcement learning [16, 17].

Yet, in safety- and reliability-critical applications, the classical CMDP viewpoint can be misaligned with how systems are deployed and assessed. Standard constraints are evaluated only from the initial state, providing guarantees tied to the beginning of the trajectory. In many settings, the continuation from time step t should satisfy the same requirements as a process starting anew. It is therefore natural to require the constraints imposed on the full process to hold for *every subprocess* following a reachable history. Without such subprocess constraints, a policy may satisfy a global expected-cost budget while behaving unacceptably after certain histories. For example, low early costs may offset high future costs under the global constraint, or high-cost trajectories may be averaged against low-cost ones.

Such concerns arise naturally in practice. For instance, when an autonomous robot reaches a crowded region or a narrow passage near humans, safety in expectation from the initial deployment state is not enough; from the newly reached configuration, the controller should still keep future collision risk below an acceptable threshold [13, 28]. Similarly, in online services governed by service-level

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

objectives (SLOs), reliability is monitored continuously through service-level indicators and error budgets. Exhausting an error budget can trigger responses such as reliability work or deployment freezes [7, 8]. It is unacceptable to enter a congested state from which expected future latency or availability violations exceed acceptable thresholds, even if the initial expected violations remain within budget. In these settings, constraints that apply only at initialization can be too weak.

1.1 Our Work and Contributions

To address this limitation, we introduce *subprocess-constrained MDPs* (S-CMDPs). Informally, S-CMDPs require policies to remain feasible under a "restart at time t" interpretation after every reachable history, capturing a subprocess-level notion of safety and reliability not enforced by classical CMDPs. Although this may appear to be a modest change in formulation, it fundamentally changes the problem: standard CMDP solution methods do not transfer directly.

We study the algorithmic aspects of S-CMDPs and establish the following foundational results:

- We identify **a fundamental computational separation** between S-CMDPs and standard CMDPs. Stationary policies, which suffice for optimality and can be optimized efficiently in classical MDPs and CMDPs, can be strictly suboptimal in S-CMDPs. More importantly, even if we accept this loss of optimality and restrict attention to stationary policies, finding the best policy in that class, even just approximately, is NP-hard.
- We show that history-dependent policies, which can be necessary for optimality, can be handled efficiently using **value-set iteration (VSI)**. This is surprising because representing a history-dependent policy explicitly requires space exponential in the horizon. In S-CMDPs, allowing history dependence is key to establishing **tractability**. Our approach leverages an implicit representation of history-dependent policies: rather than explicitly specifying action distributions, the policy is given by an algorithm that efficiently computes its action distribution for any given history.
- VSI extends classical value iteration to a *set-valued* dynamic programming framework. Unlike classical value iteration, the associated set-valued operator is generally not a contraction, so convergence requires a different analysis. We characterize the solution of an S-CMDP as a **fixed point** of this operator and develop a **bicriteria approximation scheme** that computes a fixed point of an approximate operator in polynomial time. This yields an (ϵ, δ)-optimal history-dependent policy, in contrast to the NP-hardness of computing such a stationary policy.

Our results establish S-CMDPs as a distinct extension of CMDPs that enforces a stronger notion of safety while remaining tractable through our VSI framework.

1.2 Related Work

Our starting point is classical CMDPs, which constrain expected cumulative cost from the initial state and admit methods based on linear programming and Lagrangian optimization [2, 3]. This perspective also underlies much of safe RL: constrained policy optimization targets global constraints [1], and surveys emphasize initial-state expectation constraints [16, 17]. Because a guarantee from the start need not hold after a policy reaches an intermediate state, S-CMDPs require onward feasibility after every reachable history.

Several lines of work strengthen safety requirements at individual times or states. Peak-constrained CMDPs impose constraints at every time step with probability one [5], while related safe RL formulations rule out unsafe actions or states through instantaneous hard constraints [4, 23, 26]. Other approaches constrain instantaneous violations in expectation [20] or impose state-wise conditions [27, 24]. These formulations address local or time-indexed risks; our constraints evaluate the entire discounted continuation from each reachable history.

Anytime-constrained MDPs are closer in motivation: they require a cumulative cost budget to be respected throughout the trajectory [21]. This makes accumulated cost relevant to future decisions and motivates augmenting the state with that information. S-CMDPs instead bound the expected onward cost of every subprocess, leading us to reason over sets of attainable continuation values.

On the algorithmic side, value-set iteration has precedents in set-valued dynamic programming for equilibria in sequential games [22, 10, 19, 15, 14]. In those settings, incentive constraints must hold

in every subgame, giving rise to a recursive structure similar to our subprocess constraints. The recent work of Gan and Majumdar [14] provides a detailed algorithmic study of value-set iteration for computing extensive-form correlated equilibria in Markov games. Our work adapts this framework to S-CMDPs, a setting that we believe admits an even broader range of applications in planning, control, and RL. Our VSI framework also rests on a fixed point of a monotone operator, connecting it to algorithmic work on Tarski fixed points [11, 12, 9, 6]. An extended discussion of related work can be found in Appendix A.

2 The S-CMDP Model

An S-CMDP is given by a tuple $(S, A, s_0, p, r, c, \theta, \gamma)$. The components other than c and θ define a standard discounted MDP: S is a finite state space, with initial state $s_0 \in S$; A is a finite action space; $p : S \times A \to \Delta(S)$ is the transition kernel; $r : S \times A \to \mathbb{R}$ is the reward function; and $\gamma \in (0, 1)$ is the discount factor. The additional components c and θ specify a cost function $c : S \times A \to \mathbb{R}$ and a state-wise threshold $\theta(s) \in \mathbb{R}$ for each $s \in S$. Without loss of generality, we assume that rewards and costs lie in $[-1, 1]$.

Let a policy π assign an action distribution $\pi(h) \in \Delta(A)$ to each finite history $h = (s_0, a_0, \ldots, s_t)$. For a reachable history h, we define the onward reward value as

$$V_r^\pi(h) = \mathbb{E}_\pi \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \cdot r(s_{t'}, a_{t'}) \,\middle|\, h \right],$$

where $\mathbb{E}_\pi[\cdot \mid h]$ denotes expectation conditional on h under π, and $\mathbb{P}_\pi(h)$ denotes the probability of observing h. Similarly, $V_c^\pi(h)$ denotes the onward discounted cost, defined by replacing r with c.

Our objective is to maximize expected discounted reward while bounding the onward discounted cost after every reachable history:

$$\max_\pi \quad V_r^\pi(s_0) \tag{1}$$

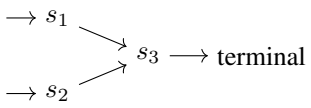
$$\text{s.t.} \quad V_c^\pi(h) \le \theta(s_t) \qquad \text{for all } h = (s_0, a_0, \ldots, s_t),\ t \ge 0 : \mathbb{P}_\pi(h) > 0 \tag{2}$$

The formulation imposes a constraint on every subprocess starting from a reachable history h.

3 Why Not Stationary Policies?

In standard CMDPs, *stationary* policies—which map each state to an action distribution—suffice for optimality. This is no longer the case for S-CMDPs. The maximum value attained by stationary policies can be strictly lower than that attained by history-dependent policies. Moreover, even if we accept this loss of optimality and restrict attention to stationary policies, optimizing over this class is computationally intractable. Both limitations persist in the finite-horizon setting.

Suboptimality of Stationary Policies Consider the S-CMDP shown on the right. The process starts in either s_1 or s_2 with equal probability.[1] There are two actions, a and b. All transitions are deterministic, as depicted in the figure, and are independent of the chosen action. All rewards and costs are zero except for $r(s_3, b) = c(s_3, b) = 1$. The cost thresholds are $\theta(s_1) = 0$ and $\theta(s_2) = \theta(s_3) = 1$.



The only way to obtain positive reward in this instance is to choose action b at s_3, which also incurs cost 1. Under any stationary policy, however, choosing b at s_3 with positive probability would violate the subprocess constraint at s_1, where the onward cost must not exceed $\theta(s_1) = 0$. Hence, every feasible stationary policy must choose action a at s_3 with probability 1, yielding value zero.[2]

In contrast, history dependence allows the agent to condition its action at s_3 on the preceding state: choose a if the trajectory starts from s_1, and choose b if it starts from s_2. This satisfies all subprocess constraints and yields expected value $\gamma/2$, strictly outperforming all feasible stationary policies.

[1] For ease of exposition, we slightly deviate from the model in Section 2 and allow a randomized initial state.

[2] In this example, even allowing the policy to be time-dependent does not improve the outcome.

Intractability of Stationary Policies A further limitation of stationary policies is computational intractability. Optimizing over feasible stationary policies, even approximately, is NP-hard. This intractability persists even when the subprocess constraints are relaxed to permit small violations. We formalize this claim with the following approximation notion and Theorem 1.

Definition 1 ((ϵ, δ)-optimal policy)**.** A policy π is (ϵ, δ)-optimal if the following conditions hold:

1) δ**-feasibility:** π violates every subprocess constraint in Eq. (2) by at most δ.

2) ϵ**-optimality:** $V_r^{\pi}(s_0) \geq V_r^{\pi'}(s_0) - \epsilon$ for every policy π' in the class under consideration that satisfies Eq. (2) exactly.

Unless stated otherwise, the class contains all policies. In Theorem 1, both the computed policy and the policies π' used for comparison are stationary.

Theorem 1. *Fix any discount factor $\gamma \in (0, 1)$. There exist constants $\epsilon, \delta > 0$ such that computing a* ***stationary*** *policy that is (ϵ, δ)-optimal among stationary policies is NP-hard.*

For the remainder of the paper, we turn to history-dependent policies and show that they overcome both limitations of stationary policies.

4 Enabling History Dependence: Two Birds with One Stone

While allowing history dependence removes the restriction behind the preceding suboptimality example, its computational tractability is far from obvious. Because the number of reachable histories can grow exponentially with the horizon, even writing down a history-dependent policy may be infeasible, let alone computing an optimal one.

We therefore adopt an implicit computational view and ask: *given any history h, can we efficiently compute the action distribution of a near-optimal policy at h?* An algorithm that does so allows us to execute the policy without explicitly storing all action distributions; effectively, it encodes the policy.

The value-set iteration (VSI) framework provides an answer to this question. In this section, we first introduce a central concept underlying the VSI framework—the *achievable value sets*—and explain how it enables us to handle history-dependent policies.

4.1 Achievable Value Sets

Our goal is to characterize which combinations of cumulative reward and cost can be attained by feasible policies. To this end, we view each state–action pair as producing a two-dimensional payoff whose components are reward and cost. Formally, define a payoff function $\mathbf{x} : S \times A \to \mathbb{R}^2$ such that

$$\mathbf{x}(s, a) := (r(s, a),\, c(s, a))$$

for every $(s, a) \in S \times A$. Similarly, for any policy π and history h, we write $\vec{V}^{\pi}(h) = (V_r^{\pi}(h), V_c^{\pi}(h))$ for the vector of onward discounted reward and cost.

Definition 2 (Achievable value set)**.** The *achievable value set* of a state s, denoted $\mathcal{V}^{\star}(s)$, is the set of all vectors $\mathbf{v} \in \mathbb{R}^2$ that can be achieved by some *feasible* policy π when starting from s, i.e., $\mathbf{v} = \vec{V}^{\pi}(s)$ for some π satisfying Eq. (2).

We call functions like $\mathcal{V}^{\star}$, that map each state to a subset of $\mathbb{R}^2$, *value-set functions*.

Using $\mathcal{V}^{\star}$, we can express the optimal objective value in Eqs. (1) and (2) as the maximum reward component over the achievable set at s_0:

$$\max_{\mathbf{v}=(v_1,v_2)\in\mathcal{V}^{\star}(s_0)} v_1.$$

If $\mathcal{V}^{\star}$ were known, this maximization would give the optimal value immediately. To further obtain a policy that achieves the optimal value, we next introduce a procedure that unrolls any target value $\mathbf{v} \in \mathcal{V}^{\star}(s)$ into a feasible policy π with $\vec{V}^{\pi}(s) = \mathbf{v}$. We will return to the computation of $\mathcal{V}^{\star}$ and address the remaining algorithmic challenges in the following sections.

4.2 Unrolling Policies Based on Achievable Sets

Consider an arbitrary state s and a target value vector $\mathbf{v} = (v_1, v_2) \in \mathcal{V}^\star(s)$. By definition, some feasible policy π achieves $\mathbf{v}$, so a Bellman-style expansion gives

$$\mathbf{v} = \mathbb{E}_{a\sim\bar{\pi}} \Big(\mathbf{x}(s,a) + \gamma\, \mathbb{E}_{s'\sim p(\cdot \mid s,a)}\, \mathbf{w}(a,s') \Big). \tag{3}$$

Here $\bar{\pi} = \pi(s) \in \Delta(A)$ is the action distribution at state s; and for each pair (a, s'), $\mathbf{w}(a,s') = \vec{V}^\pi(s,a,s')$ is the onward value, given history (s,a,s').

Crucially, the fact that each onward value $\mathbf{w}(a,s')$ is achieved in the subsequent process means that:

$$\mathbf{w}(a,s') \in \mathcal{V}^\star(s') \;\text{ for all } a, s' : \bar{\pi}(a)\cdot p(s' \mid s,a) > 0, \tag{4}$$

where only the (s,a,s') triples with positive probability under $\bar{\pi}$ and the transition function p are relevant, as the others do not contribute to Eq. (3).

Policy Execution via Recursive Value Expansion Eqs. (3) and (4) lead to the recursive value expansion procedure in Alg. 2. The procedure expands a given target value $\mathbf{v} \in \mathcal{V}^\star(s)$ into an action distribution $\bar{\pi}$ and a collection $\mathbf{w}$ of onward values based on Eqs. (3) and (4). It samples an action a from $\bar{\pi}$ and observes the next state s'. The onward value $\mathbf{w}(a,s')$ is then chosen as the new target, and the procedure recurses. We fix a deterministic rule for selecting a valid decomposition whenever several are available.

Since $\mathbf{v} \in \mathcal{V}^\star(s)$, there always exists a valid decomposition into $\bar{\pi}$ and $\mathbf{w}$. Moreover, Eq. (4) ensures that the new target value also lies within the corresponding achievable set, so a valid decomposition exists at the next recursive call as well. Thus, at every recursive call with current state s, the target value lies in $\mathcal{V}^\star(s)$, so the procedure can continue indefinitely.

`Expand`$(\mathbf{v},\, s)$**:**
- **Return** $(\bar{\pi}, \mathbf{w})$ satisfying Eqs. (3) and (4).

Alg 1. `Expand`.

`Achieve`$(\mathbf{v},\, s)$**:**
1. $(\bar{\pi}, \mathbf{w}) \leftarrow$ `Expand`$(\mathbf{v}, s)$.
2. Sample $a \sim \bar{\pi}$; observe $s' \sim p(\cdot \mid s, a)$.
3. `Achieve`$(\mathbf{v}', s')$, where $\mathbf{v}' = \mathbf{w}(a, s')$.

Alg 2. `Achieve`.

We can verify that the recursive procedure has expected discounted reward and cost exactly equal to $\mathbf{v}$. Effectively, the procedure implements a policy that achieves $\mathbf{v}$ (i.e., $\vec{V}^\pi(s) = \mathbf{v}$).

Notably, executing this policy requires storing only the current target value. At each step, the target is expanded into 1) an action distribution used to sample the next action and 2) onward values used to update the target. Thus, the target serves as a fixed-dimensional summary of the history, independent of the horizon length.[3]

Computing the Policy Returning to the algorithmic question posed at the beginning of this section, we can adapt the recursive expansion procedure to compute the action distribution $\pi(h)$ for any history h starting at s_0 and any target $\mathbf{v} \in \mathcal{V}^\star(s_0)$, where π is a policy that achieves $\mathbf{v}$. We present this algorithm as `ActProbs` in Alg. 3.

`ActProbs`$(\mathbf{v},\, h = (s_0, a_0, \ldots, s_t))$**:**
1. $(\bar{\pi}, \mathbf{w}) \leftarrow$ `Expand`$(\mathbf{v}, s_0)$.
2. If $t = 0$, then **return** $\bar{\pi}$.
3. **Return** `ActProbs`$(\mathbf{v}',\, h')$, where $\mathbf{v}' = \mathbf{w}(a_0, s_1)$ and $h' = (s_1, a_1, \ldots, s_t)$.[4]

Alg 3. `ActProbs`.

Rather than sampling actions and states, `ActProbs` follows the given history h. On reachable histories, its recursive calls return the action distribution induced by `Achieve`$(\mathbf{v}, s_0)$ conditional on h.

Let π be the policy computed by `ActProbs`$(\mathbf{v}, \cdot)$, i.e., $\pi(h) \coloneqq$ `ActProbs`$(\mathbf{v}, h)$ for every history h starting at s_0. It can be verified that π is feasible and $\vec{V}^\pi(s_0) = \mathbf{v}$. Hence, choosing $\mathbf{v}^* \in \arg\max_{\mathbf{v}\in\mathcal{V}^\star(s_0)} v_1$ as the target yields an optimal policy for the S-CMDP.

[3] The precision required to represent the exact target value may grow over time; the approximation approach in Section 6 addresses this issue.

[4] If $\bar{\pi}(a_0)\, p(s_1 \mid s_0, a_0) = 0$ at any recursive step, then the input history is unreachable under the induced policy. In this case, `ActProbs` returns an arbitrary fixed distribution in $\Delta(A)$ without evaluating $\mathbf{w}(a_0, s_1)$.

The following sections formalize these claims. In summary, the algorithmic challenge of S-CMDPs now reduces to: 1) computing $\mathcal{V}^\star$, and 2) computing $\bar{\pi}$ and $\mathbf{w}$ that satisfy Eqs. (3) and (4). We introduce the VSI algorithm next.

5 Value-Set Iteration

At a high level, our approach mirrors classical value iteration for unconstrained MDPs: we define an operator whose *fixed point* characterizes the achievable values, and we design an iterative process that converges to this fixed point. In our setting, however, the operator acts on sets of vectors rather than scalar values. More crucially, due to the subprocess constraints, its convergence requires a fundamentally different argument from that used for classical value iteration.

5.1 $\mathcal{V}^\star$ as a Fixed Point

To begin, let us characterize $\mathcal{V}^\star$ as a fixed point. Recall from Eqs. (3) and (4) that every $\mathbf{v} \in \mathcal{V}^\star(s)$ admits a Bellman-style decomposition. In our S-CMDP model, there is another necessary condition: at state s, the cost component of every achievable vector $\mathbf{v}$ must not exceed the state-wise threshold:

$$v_2 \leq \theta(s). \tag{5}$$

This condition is not just necessary: together with Eqs. (3) and (4), it is sufficient for the achievability of $\mathbf{v}$, as the following lemma states.

Lemma 2. *For any $s \in S$ and $\mathbf{v} = (v_1, v_2) \in \mathbb{R}^2$, we have $\mathbf{v} \in \mathcal{V}^\star(s)$ if and only if $v_2 \leq \theta(s)$ and there exist a distribution $\bar{\pi} \in \Delta(A)$ and a vector $\mathbf{w}(a, s') \in \mathbb{R}^2$ for each $(a, s') \in A \times S$ such that Eqs.* (3) *and* (4) *hold.*

Lemma 2 characterizes $\mathcal{V}^\star$ as a fixed point of the following operator Φ on value-set functions. For any $s \in S$,

$$\Phi(\mathcal{V})(s) := \left\{ \mathbf{v} \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} \mathbf{v} = \mathrm{Bellman}(s, \bar{\pi}, \mathbf{w}), \\ (\bar{\pi}, \mathbf{w}) \in \mathrm{Onward}(s, \mathcal{V}), \\ v_2 \leq \theta(s) \end{array} \right\}, \tag{6}$$

where we use the shorthand $\mathrm{Bellman}(s, \bar{\pi}, \mathbf{w})$ to represent the right-hand side of Eq. (3), and

$$\mathrm{Onward}(s, \mathcal{V}) := \{(\bar{\pi}, \mathbf{w}) \mid \ \mathbf{w}(a, s') \in \mathcal{V}(s') \ \text{ if } \ \bar{\pi}(a) \cdot p(s' \mid s, a) > 0 \ \} \tag{7}$$

consists of all $(\bar{\pi}, \mathbf{w})$ pairs satisfying the onward achievability condition for every $(a, s') \in A \times S$, with respect to $\mathcal{V}$ (i.e., Eq. (4) with $\mathcal{V}^\star$ replaced by $\mathcal{V}$).

Intuitively, $\Phi(\mathcal{V})(s)$ consists of values obtained by combining an action distribution with onward values from $\mathcal{V}$, subject to the cost threshold at s. Lemma 2 immediately yields the following corollary.

Corollary 3. *$\mathcal{V}^\star$ is a fixed point of Φ, i.e., $\mathcal{V}^\star = \Phi(\mathcal{V}^\star)$.*

5.2 Convergence of Value-Set Iteration to $\mathcal{V}^\star$

This suggests computing a fixed point of Φ to obtain $\mathcal{V}^\star$. For discounted unconstrained MDPs, the Bellman optimality operator admits a unique fixed point, which is exactly the optimal value function. The existence and uniqueness of this fixed point follow from the contraction mapping theorem (Banach's fixed-point theorem), thanks to the Bellman operator being a *contraction*.[5] Thus, through repeated application of the Bellman operator, classical value iteration converges to the fixed point at a geometric rate.

`VSI`$(\Phi, \mathcal{V})$**:**
1. Repeat $\mathcal{V} \leftarrow \Phi(\mathcal{V})$ until $\mathcal{V} = \Phi(\mathcal{V})$.
2. **Return** $\mathcal{V}$.

Alg 4. `VSI` (value-set iteration).

Motivated by the above, it is tempting to design a similar iteration scheme by repeatedly applying Φ to a value-set function (Alg. 4), in the hope that it converges to $\mathcal{V}^\star$.

[5] A mapping T on a metric space $(\mathcal{X}, d)$ is a contraction if there exists $\beta \in [0, 1)$ such that $d(T(x), T(y)) \leq \beta\, d(x, y)$ for all $x, y \in \mathcal{X}$.

This approach, however, requires more care. Unlike the classical Bellman operator, Φ need not be a contraction, so the contraction mapping theorem does not apply in general. Its fixed point may not be unique;[6] moreover, repeated application of Φ from an arbitrary initial value-set function may fail to converge or may converge to an undesired fixed point.

As it turns out, the following *monotonicity* property of Φ becomes key to our convergence argument.

Lemma 4 (Monotonicity)**.** *If $\mathcal{V} \subseteq \mathcal{V}'$, then $\Phi(\mathcal{V}) \subseteq \Phi(\mathcal{V}')$.*[7]

Convergence by Monotonicity The monotonicity property motivates a fixed-point argument inspired by Tarski's fixed-point theorem [25]. The theorem states that the fixed points of a monotone self-map on a complete lattice form a complete lattice and, in particular, include a *greatest* fixed point. In our setting, $\mathcal{V}^\star$ is precisely the greatest bounded fixed point of Φ: it contains every bounded fixed point of Φ (as we establish directly in Lemma 7).

More importantly, monotonicity suggests a way to find this fixed point. If we initialize VSI (Alg. 4) from an upper bound $\mathcal{V}$ of $\mathcal{V}^\star$ that Φ maps into itself, i.e., $\Phi(\mathcal{V}) \subseteq \mathcal{V}$, we obtain a decreasing sequence of value-set functions. Indeed, since all rewards and costs lie in $[-1, 1]$, the L_∞ ball

$$\mathcal{B} := \left\{ \mathbf{v} \in \mathbb{R}^2 \;\middle|\; \|\mathbf{v}\|_\infty \leq \tfrac{1}{1-\gamma} \right\}$$

provides such an upper bound, as the following lemma states.

Lemma 5. *$\mathcal{V}^\star \subseteq \mathcal{B}$ and $\Phi(\mathcal{B}) \subseteq \mathcal{B}$.*[8]

We show that the iterates of $\texttt{VSI}(\Phi, \mathcal{B})$ converge to $\mathcal{V}^\star$. Let $(\mathcal{V}^k)_{k\geq 0}$ be the sequence it generates:

$$\mathcal{V}^0 = \mathcal{B}, \quad \text{and} \quad \mathcal{V}^{k+1} = \Phi(\mathcal{V}^k) \text{ for all } k \geq 0.$$

By Lemma 5, $\mathcal{V}^\star \subseteq \mathcal{B}$, so monotonicity of Φ gives $\Phi(\mathcal{V}^\star) \subseteq \Phi(\mathcal{B})$. Since $\mathcal{V}^\star = \Phi(\mathcal{V}^\star)$ and $\Phi(\mathcal{B}) \subseteq \mathcal{B}$, we obtain

$$\mathcal{V}^\star \subseteq \Phi(\mathcal{B}) \subseteq \mathcal{B}.$$

Recall that $\mathcal{V}^0 = \mathcal{B}$ and $\mathcal{V}^1 = \Phi(\mathcal{V}^0)$, so this becomes

$$\mathcal{V}^\star \subseteq \mathcal{V}^1 \subseteq \mathcal{V}^0.$$

By monotonicity, repeatedly applying Φ to these inclusions propagates the nesting to all k, yielding the following result.

Lemma 6. *$\mathcal{V}^\star \subseteq \mathcal{V}^{k+1} \subseteq \mathcal{V}^k$ for all $k \geq 0$.*

Consequently, $(\mathcal{V}^k)_{k\geq 0}$ is a decreasing sequence bounded below by $\mathcal{V}^\star$. Each $\mathcal{V}^k(s)$ is compact, and a compactness argument shows that the intersection of these iterates is a fixed point of Φ (see the proof of Theorem 8). This fixed point must equal $\mathcal{V}^\star$, which bounds it not only from below but also, by the following lemma, from above.

Lemma 7. *If $\mathcal{V}$ is bounded and $\mathcal{V} = \Phi(\mathcal{V})$, then $\mathcal{V} \subseteq \mathcal{V}^\star$.*

The proof of Lemma 7 follows the recursive construction in Section 4, where we showed that every $\mathbf{v} \in \mathcal{V}^\star(s)$ can be achieved by the `Achieve` procedure (Alg. 2). Here, we show more generally that for any bounded fixed point $\mathcal{V}$ of Φ, every $\mathbf{v} \in \mathcal{V}(s)$ can be achieved by the same procedure, with $\mathcal{V}$ in place of $\mathcal{V}^\star$ in the onward value constraint in Eq. (4). This modified procedure is well defined because $\mathcal{V} = \Phi(\mathcal{V})$ guarantees a valid decomposition at every recursive call. Consequently, every $\mathbf{v} \in \mathcal{V}(s)$ is realized as the expected discounted return of a feasible policy and is therefore achievable from s.

These lemmas lead to the following convergence theorem.

Theorem 8. *$\mathcal{V}^k \to \mathcal{V}^\star$ as $k \to \infty$.*[9]

[6] For example, assigning the empty set to every state gives a trivial fixed point, distinct from $\mathcal{V}^\star$ whenever a feasible policy exists from at least one state.

[7] For simplicity, we use $\subseteq$ to denote the pointwise inclusion: e.g., $\mathcal{V} \subseteq \mathcal{V}'$ when $\mathcal{V}(s) \subseteq \mathcal{V}'(s)$ for all $s \in S$.

[8] When we apply Φ to $\mathcal{B}$, we interpret $\mathcal{B}$ as the constant value-set function mapping every state to the ball $\mathcal{B}$.

[9] More precisely, we have $\mathcal{V}^\infty = \mathcal{V}^\star$, where $\mathcal{V}^\infty(s) := \bigcap_{k\geq 0} \mathcal{V}^k(s)$ for every $s \in S$.

6 Approximate VSI for Finite-Time Convergence

We have established the convergence of value-set iteration to $\mathcal{V}^\star$, but this does not automatically yield a finite-time algorithm, let alone an efficient one. To obtain a practical algorithm, two issues must be addressed: 1) ensuring termination in finitely—preferably polynomially—many iterations; and 2) representing value-set functions in a way that allows applications of Φ to be computed efficiently.

As in practical implementations of value iteration, we adopt an approximation approach. Rather than computing $\mathcal{V}^\star$ exactly, we iterate an approximate operator that reaches a fixed point in finitely many steps. As we show in this section, this approach is sufficient for our purpose of computing an (ϵ, δ)-optimal policy (Definition 1).

6.1 Approximate Operator and Convergence

Starting from $\mathcal{B}$, repeated applications of Φ produce convex polytopes at each state (as the LP formulation in Section 6.3 makes explicit). These convex polytopes can in principle be represented by linear inequalities, but the issue is that the number of inequalities can grow exponentially under repeated applications of Φ. To control the representation complexity, we discretize the two-dimensional value space using a grid and approximate Φ based on this discretization.

Approximate Operator Choose a positive grid spacing $\xi \leq (1-\gamma) \cdot \min\{\epsilon, \delta/2\}$. Let $G_\xi := \{\xi \mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^2\}$ be the set of grid points. For any $X \subseteq \mathbb{R}^2$, define

$$\mathrm{GridHull}_\xi(X) := \mathrm{conv}(X \cap G_\xi)\,,$$

where $\mathrm{conv}(\cdot)$ denotes the convex hull. Define the approximate operator as

$$\widehat{\Phi}(\mathcal{V})(s) := \mathrm{GridHull}_\xi\big(\mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))\big)\,, \qquad (8)$$

where $\mathrm{neigh}_\xi(X)$ denotes the ξ-neighborhood of X, consisting of all points within L_∞ distance ξ of some point in X.

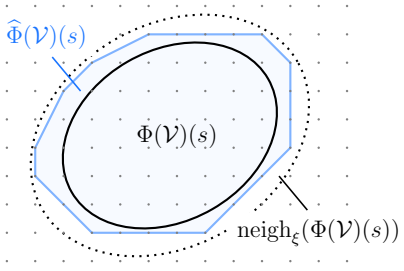


Figure 1: Illustration of $\widehat{\Phi}$.

Figure 1 illustrates this construction. Whenever $\Phi(\mathcal{V})(s)$ is convex, its ξ-neighborhood is also convex, and we can verify that the following inclusions hold:

$$\Phi(\mathcal{V})(s) \subseteq \widehat{\Phi}(\mathcal{V})(s) \subseteq \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))\,.$$

Specifically, the left-hand inclusion holds even without convexity of $\Phi(\mathcal{V})(s)$: every point of $\Phi(\mathcal{V})(s)$ lies in a grid cell whose vertices are within L_∞ distance ξ of that point; the cell is contained entirely in $\mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$ as well as its grid hull. The right-hand inclusion follows because the convex neighborhood contains the convex hull of its grid points.

Finite-Time Convergence The operator $\widehat{\Phi}$ remains monotone. To initialize VSI, let $\widehat{\mathcal{B}}_\xi$ denote the smallest grid-aligned L_∞ ball that *strictly* contains $\mathcal{B}$, i.e.,

$$\widehat{\mathcal{B}}_\xi := \left\{ \mathbf{v} \in \mathbb{R}^2 \;\middle|\; \|\mathbf{v}\|_\infty \leq \left(\left\lfloor \tfrac{1}{1-\gamma} / \xi \right\rfloor + 1 \right) \cdot \xi \right\}.$$

We run $\texttt{VSI}(\widehat{\Phi}, \widehat{\mathcal{V}}^0)$ (Alg. 4) with $\widehat{\mathcal{V}}^0 = \widehat{\mathcal{B}}_\xi$. This yields a sequence $(\widehat{\mathcal{V}}^k)_{k \geq 0}$ such that:

$$\widehat{\mathcal{V}}^0 = \widehat{\mathcal{B}}_\xi, \quad \text{and} \quad \widehat{\mathcal{V}}^{k+1} = \widehat{\Phi}(\widehat{\mathcal{V}}^k) \text{ for all } k \geq 0.$$

The strict outer rounding of $\widehat{\mathcal{V}}^0$ ensures $\widehat{\Phi}(\widehat{\mathcal{V}}^0) \subseteq \widehat{\mathcal{V}}^0$. Together with $\mathcal{V}^\star \subseteq \mathcal{B} \subseteq \widehat{\mathcal{V}}^0$ and the monotonicity of $\widehat{\Phi}$, this makes $(\widehat{\mathcal{V}}^k)_k$ a decreasing sequence of nested value sets. We extend Lemma 6 to this sequence as follows.

Lemma 9. *$\mathcal{V}^\star \subseteq \widehat{\mathcal{V}}^{k+1} \subseteq \widehat{\mathcal{V}}^k$ for all $k \geq 0$.*

Given Lemma 9, the discretization then ensures finite-time convergence. Specifically, by construction, each $\widehat{\mathcal{V}}^k(s)$ is fully characterized by the grid points it contains. Since the sequence is decreasing, whenever $\widehat{\mathcal{V}}^{k+1} \neq \widehat{\mathcal{V}}^k$, at least one grid point must have been removed from the latter. Let $N_\xi = |G_\xi \cap \widehat{\mathcal{B}}_\xi|$ be the number of grid points inside $\widehat{\mathcal{B}}_\xi$. Across the $|S|$ value sets, at most $|S|N_\xi$ grid points can be removed, so $\texttt{VSI}(\widehat{\Phi}, \widehat{\mathcal{V}}^0)$ reaches a fixed point within $O(|S|N_\xi)$ iterations. When γ and ξ are fixed, N_ξ is a constant, so the iteration bound is $O(|S|)$.

6.2 (ϵ, δ)-Optimality at Convergence

Let $\widehat{\mathcal{V}}^\star$ be the fixed point obtained via $\texttt{VSI}(\widehat{\Phi}, \widehat{\mathcal{V}}^0)$, i.e., $\widehat{\mathcal{V}}^\star = \widehat{\mathcal{V}}^k = \widehat{\mathcal{V}}^{k+1}$ for some $k \geq 0$. By Lemma 9, $\mathcal{V}^\star \subseteq \widehat{\mathcal{V}}^\star$, so the approximation retains all achievable values, including the optimal one.

Although $\widehat{\mathcal{V}}^\star$ may contain values outside $\mathcal{V}^\star$, Lemma 10 shows that every value it contains is achievable *approximately*, in the sense defined below. Choose $\mathbf{v}^* \in \arg\max_{\mathbf{v} \in \widehat{\mathcal{V}}^\star(s_0)} v_1$. The lemma gives a δ-feasible policy whose return is within ϵ of $\mathbf{v}^*$. Since v_1^* is at least the reward of every exactly feasible policy, this policy is (ϵ, δ)-optimal (Definition 1).

Definition 3 ((ϵ, δ)-achievability)**.** $\mathbf{v} \in \mathbb{R}^2$ is (ϵ, δ)-*achievable* from state s if there exists a δ-feasible policy π, with $\|\vec{V}^\pi(s) - \mathbf{v}\|_\infty \leq \epsilon$.

Lemma 10. *If $\mathcal{V}$ is bounded and $\mathcal{V} = \widehat{\Phi}(\mathcal{V})$, then for all $s \in S$ every $\mathbf{v} \in \mathcal{V}(s)$ is (ϵ, δ)-achievable from s.*

The proof of Lemma 10 follows the same recursive construction as that of Lemma 7, allowing a bounded error at each expansion. The corresponding extension of `Achieve` (Alg. 2), denoted $\texttt{Achieve}_\xi$, replaces `Expand` with the relaxed subroutine $\texttt{Expand}_\xi$ shown in Alg. 5.

Relaxed Value Expansion Compared with the exact implementation, $\texttt{Expand}_\xi$ only requires $(\bar{\pi}, \mathbf{w})$ to induce a value close to $\mathbf{v}$. This relaxation is essential because $\widehat{\mathcal{V}}^\star$ is a fixed point of $\widehat{\Phi}$, but need not be a fixed point of the original operator Φ. In general, it can contain values outside $\Phi(\widehat{\mathcal{V}}^\star)$.

$\texttt{Expand}_\xi(\mathbf{v},\ s)$**:**
> **Return** $(\bar{\pi}, \mathbf{w}) \in \mathrm{Onward}(s, \widehat{\mathcal{V}}^\star)$ such that $\mathrm{Bellman}(s, \bar{\pi}, \mathbf{w}) \in \mathrm{neigh}_\xi(\mathbf{v})$.

Alg 5. $\texttt{Expand}_\xi$.

Indeed, insisting on an exact Bellman expansion of such values may require onward vectors outside $\widehat{\mathcal{V}}^\star$. Repeating this at subsequent steps can cause the targets to drift outward and become unbounded, preventing them from being realized as discounted returns.

The procedure $\texttt{Expand}_\xi$ prevents this drift by keeping the onward vectors within $\widehat{\mathcal{V}}^\star$ while allowing a Bellman error of at most ξ at each step. Such an expansion exists because $\widehat{\mathcal{V}}^\star \subseteq \mathrm{neigh}_\xi(\Phi(\widehat{\mathcal{V}}^\star))$. Due to geometric discounting, the total accumulated offset is bounded by $\xi/(1-\gamma) \leq \epsilon$.

Replacing `Expand` with $\texttt{Expand}_\xi$ in Alg. 3 yields a corresponding extension of `ActProbs`. The extension, call it $\texttt{ActProbs}_\xi$, computes the action distributions prescribed by a δ-feasible policy that achieves the target value within error ϵ. We omit the pseudocode for both $\texttt{ActProbs}_\xi$ and $\texttt{Achieve}_\xi$.

6.3 Computation and Tractability

It remains to make these operations computationally efficient. As outlined in Section 4, we need to:

1) compute applications of $\widehat{\Phi}$, so that we can obtain its fixed point using `VSI`; and
2) compute $(\bar{\pi}, \mathbf{w})$ in $\texttt{Expand}_\xi$, so that ultimately we can execute an (ϵ, δ)-optimal policy using $\texttt{Achieve}_\xi$, or compute its action distributions using $\texttt{ActProbs}_\xi$.

We establish tractability by reducing both tasks to linear programming (LP).

Computing $\widehat{\Phi}(\mathcal{V})(s)$ From the construction in Eq. (8), computing each $\widehat{\Phi}(\mathcal{V})(s)$ reduces to determining, for each grid point $\mathbf{g}$, whether $\mathbf{g} \in \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$. We can restrict attention to grid points in $\widehat{\mathcal{B}}_\xi$ because by Lemma 9 all relevant value sets are contained in $\widehat{\mathcal{V}}^0 = \widehat{\mathcal{B}}_\xi$. Thus, for fixed γ and ξ, there is only a fixed number of grid points to check per state.

To efficiently decide whether $\mathbf{g} \in \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$, we show that $\mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$ can be characterized by linear constraints, so the decision problem reduces to an LP. Indeed, $\mathbf{v} \in \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$ means that there exists some $\tilde{\mathbf{v}} \in \Phi(\mathcal{V})(s)$ such that

$$\|\mathbf{v} - \tilde{\mathbf{v}}\|_\infty \leq \xi. \tag{9}$$

1. Run $\texttt{VSI}(\widehat{\Phi}, \widehat{\mathcal{V}}^0)$ to obtain a fixed point $\widehat{\mathcal{V}}^\star$, where $\widehat{\mathcal{V}}^0 = \widehat{\mathcal{B}}_\xi$. If $\widehat{\mathcal{V}}^\star(s_0) = \varnothing$, report that no exactly feasible policy exists and stop.
2. Choose a target $\mathbf{v}^* \in \arg\max_{\mathbf{v} \in \widehat{\mathcal{V}}^\star(s_0)} v_1$.
3. **To execute a policy:** Run $\texttt{Achieve}_\xi(\mathbf{v}^*,\, s_0)$ to achieve $\mathbf{v}^*$ within ϵ error while violating each subprocess constraint by at most δ.
   To compute a policy: Run $\texttt{ActProbs}_\xi(\mathbf{v}^*, h)$ to compute $\pi(h)$ of an (ϵ, δ)-optimal policy π.

Alg 6. VSI framework for solving S-CMDPs.

According to the definition of Φ in Eq. (6), the membership $\tilde{\mathbf{v}} \in \Phi(\mathcal{V})(s)$ is in turn equivalent to the existence of $\bar{\pi} \in \Delta(A)$ and $\mathbf{w}$ satisfying the following constraints:

$$\tilde{\mathbf{v}} = \sum_{a \in A} \bar{\pi}(a) \left(\mathbf{x}(s,a) + \gamma \sum_{s' \in S} p(s' \mid s, a)\, \mathbf{w}(a, s') \right), \tag{10}$$

$$\tilde{v}_2 \le \theta(s), \tag{11}$$

$$\mathbf{w}(a, s') \in \mathcal{V}(s') \qquad \text{for all } a, s' : \bar{\pi}(a) \cdot p(s' \mid s, a) > 0. \tag{12}$$

Hence, $\mathbf{v} \in \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$ if and only if there exist $\tilde{\mathbf{v}}$, $\bar{\pi}$, and $\mathbf{w}$ satisfying Eqs. (9) to (12).

These constraints are nonlinear in $\bar{\pi}$ and $\mathbf{w}$: Eq. (10) contains products $\bar{\pi}(a) \cdot \mathbf{w}(a, s')$, while the pairs constrained by Eq. (12) depend on $\bar{\pi}(a)$. To convert them into standard linear constraints, we first rewrite Eq. (12), for pairs with $p(s' \mid s, a) > 0$ and $\mathcal{V}(s') \neq \varnothing$, as

$$\bar{\pi}(a) \cdot \mathbf{w}(a, s') \;\in\; \bar{\pi}(a) \cdot \mathcal{V}(s'),$$

where $\bar{\pi}(a) \cdot \mathcal{V}(s') \coloneqq \{\bar{\pi}(a)\mathbf{v} \mid \mathbf{v} \in \mathcal{V}(s')\}$ is $\mathcal{V}(s')$ scaled by $\bar{\pi}(a)$. If $p(s' \mid s, a) > 0$ but $\mathcal{V}(s') = \varnothing$, feasibility instead requires $\bar{\pi}(a) = 0$. For nonempty successor sets, each product $\bar{\pi}(a)\mathbf{w}(a, s')$ is a convex combination of the scaled vertices. We introduce convex weights to eliminate $\mathbf{w}(a, s')$ and introduce auxiliary variables to linearize the products between $\bar{\pi}(a)$ and these weights.

More formally, we present the following lemma, establishing the tractability of $\widehat{\Phi}(\mathcal{V})(s)$.

Lemma 11. *Suppose each $\mathcal{V}(s')$ is given as the convex hull of finitely many grid points. For any s and $\mathbf{v}$, the feasibility of Eqs.* (9) *to* (12) *can be decided by solving an LP.*

Computing $\texttt{Expand}_\xi(\mathbf{v}, s)$ For any $\mathbf{v} \in \widehat{\mathcal{V}}^\star(s)$, a valid output of $\texttt{Expand}_\xi(\mathbf{v}, s)$ can be obtained by solving Eqs. (9), (10) and (12) with $\mathcal{V} = \widehat{\mathcal{V}}^\star$ and the given $\mathbf{v}$. The same linearization as above applies directly, so the tractability of $\texttt{Expand}_\xi(\mathbf{v}, s)$ follows.

Consequently, we establish Theorem 12 as a main algorithmic result. An overview of the VSI framework is given in Alg. 6.

Theorem 12. *For fixed $\gamma \in (0, 1)$, choosing $\xi = (1 - \gamma)\min\{\epsilon, \delta/2\}$ allows us to compute a fixed point $\widehat{\mathcal{V}}^\star \supseteq \mathcal{V}^\star$ of $\widehat{\Phi}$ in time polynomial in the input size, $1/\epsilon$, and $1/\delta$. If $\widehat{\mathcal{V}}^\star(s_0) \neq \varnothing$, there exists an (ϵ, δ)-optimal policy π whose action distribution $\pi(h)$ can be computed in time polynomial in the same quantities and $|h|$ for any given history h.*

7 Conclusion

We introduced S-CMDPs, which strengthen classical CMDPs by requiring cost constraints to hold for every subprocess encountered during execution. We adapted the VSI framework to show that history-dependent policies are surprisingly tractable in this setting. Besides the theoretical results, we provide a Python implementation based on MILP vertex computation; details are given in Appendix C. Two natural next steps are to consider multiple cost functions and move beyond the model-based setting. Although our algorithm extends to d cost functions by enlarging the value space from $\mathbb{R}^2$ to $\mathbb{R}^{d+1}$, the number of grid points grows exponentially in d. Developing scalable approaches for such settings—potentially combining our structural insights with heuristic or RL-based methods—remains an interesting direction.

Acknowledgments and Disclosure of Funding

J.G. was supported by the European Research Council (ERC) under the European Union's Horizon Europe programme (ASPAC, Grant Agreement No. 101223042). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[2] Eitan Altman. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical methods of operations research*, 48(3): 387–417, 1998.

[3] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

[4] Sanae Amani, Christos Thrampoulidis, and Lin Yang. Safe reinforcement learning with linear function approximation. In *International Conference on Machine Learning*, pages 243–253. PMLR, 2021.

[5] Qinbo Bai, Vaneet Aggarwal, and Ather Gattami. Provably sample-efficient model-free algorithm for MDPs with peak constraints. *Journal of Machine Learning Research*, 24(60):1–25, 2023. URL `http://jmlr.org/papers/v24/21-0117.html`.

[6] Eleni Batziou, John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Monotone contractions. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC '25)*, page 507–517, 2025.

[7] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, 2016.

[8] Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara, and Stephen Thorne. *The Site Reliability Workbook: Practical Ways to Implement SRE*. O'Reilly Media, 2018.

[9] Xi Chen and Yuhao Li. Improved upper bounds for finding tarski fixed points. In *Proceedings of the 23rd ACM Conference on Economics and Computation (EC '22)*, pages 1108–1118, 2022.

[10] Liam Dermed and Charles Isbell. Solving stochastic games. *Advances in Neural Information Processing Systems (NIPS'09)*, 22, 2009.

[11] Kousha Etessami, Christos Papadimitriou, Aviad Rubinstein, and Mihalis Yannakakis. Tarski's theorem, supermodular games, and the complexity of equilibria. *arXiv preprint arXiv:1909.03210*, 2019.

[12] John Fearnley, Dömötör Pálvölgyi, and Rahul Savani. A faster algorithm for finding tarski fixed points. *ACM Transactions on Algorithms (TALG)*, 18(3):1–23, 2022.

[13] Jaime F. Fisac, Andrea Bajcsy, Sylvia L. Herbert, David Fridovich-Keil, Steven Wang, Claire J. Tomlin, and Anca D. Dragan. Probabilistically safe robot planning with confidence-based human predictions. In *Robotics: Science and Systems*, 2018.

[14] Jiarui Gan and Rupak Majumdar. Value-set iteration: Computing optimal correlated equilibria in infinite-horizon multi-player stochastic games, 2025. URL `https://arxiv.org/abs/2506.07186`.

[15] Jiarui Gan, Rupak Majumdar, Debmalya Mandal, and Goran Radanovic. Sequential principal-agent problems with communication: Efficient computation and learning, 2023. URL `https://arxiv.org/abs/2306.03832v2`.

[16] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[17] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theories and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[18] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4): 798–859, 2001.

[19] Mitri Kitti. Subgame perfect equilibria in discounted stochastic games. *Journal of Mathematical Analysis and Applications*, 435(1):253–266, 2016.

[20] Jingqi Li, David Fridovich-Keil, Somayeh Sojoudi, and Claire J Tomlin. Augmented lagrangian method for instantaneously constrained reinforcement learning problems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2982–2989. IEEE, 2021.

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

[22] Chris Murray and Geoff Gordon. Finding correlated equilibria in general sum stochastic games. Technical Report CMU-ML-07-113, Machine Learning Department, School of Computer Science, Carnegie Mellon University, 2007.

[23] Ming Shi, Yingbin Liang, and Ness Shroff. A near-optimal algorithm for safe reinforcement learning under instantaneous hard constraints. In *International Conference on Machine Learning*, pages 31243–31268. PMLR, 2023.

[24] Yifan Sun, Feihan Li, Weiye Zhao, Rui Chen, Tianhao Wei, and Changliu Liu. Learn with imagination: Safe set guided state-wise constrained policy optimization. In Necmiye Ozay, Laura Balzano, Dimitra Panagou, and Alessandro Abate, editors, *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, volume 283 of *Proceedings of Machine Learning Research*, pages 298–309. PMLR, 04–06 Jun 2025. URL `https://proceedings.mlr.press/v283/sun25a.html`.

[25] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285—-309, 1955.

[26] Honghao Wei, Xin Liu, and Lei Ying. Safe reinforcement learning with instantaneous constraints: the role of aggressive exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21708–21716, 2024.

[27] Weiye Zhao, Rui Chen, Yifan Sun, Feihan Li, Tianhao Wei, and Changliu Liu. State-wise constrained policy optimization. *Transactions on Machine Learning Research*, 2024.

[28] Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.

A Related Work (Extended Version)

Our work builds on several strands of CMDP and safe RL research, while differing from them in its treatment of onward feasibility. Classical CMDPs constrain expected cumulative cost from the initial state and admit strong structural results and solution methods, including linear programming and Lagrangian optimization [2, 3]. Much of modern safe RL inherits this formulation; for example, constrained policy optimization (CPO) [1] targets global constraints while providing guarantees about constraint satisfaction. Surveys of safe RL [16, 17] similarly center on initial-state expectation constraints and closely related risk criteria. Our S-CMDP model imposes a stronger feasibility requirement: onward feasibility after every reachable history, rather than only feasibility in expectation from the start.

Several lines of work in safe RL and constrained MDPs strengthen safety requirements at individual times or states. For example, peak-constrained CMDPs [5] require constraints to hold at every time step with probability one; in such formulations, the constraint can often be viewed as inducing a safe action set at each state-time pair. Related safe RL formulations impose instantaneous hard constraints that rule out unsafe actions or unsafe states during learning [4, 23, 26]. A distinct line of work studies instantaneous constraints in expectation and develops Lagrangian or augmented-Lagrangian methods for enforcing time-indexed constraint violations [20]. Recent work on state-wise constrained policy optimization also imposes constraints at individual states [27], including variants that guide learning by requiring visited states to remain within a safe state set [24].

These formulations are complementary in motivation, but they primarily constrain local quantities: immediate action safety, next- or visited-state safety, worst-case state-wise violations, or expected instantaneous violations. By contrast, S-CMDPs constrain the entire discounted continuation from each reachable history, which spans multiple or even infinite time steps rather than only the next action, transition, or visited state.

Closer to our model is the recent work on anytime-constrained MDPs [21]. Anytime-constrained MDPs also recognize that feasibility from the initial state alone may be insufficient, and instead require the agent to avoid violating its budget at any time. McMahan and Zhu [21] show, among other results, that Markovian policies are generally insufficient unless augmented with additional information, such as cumulative cost. Our S-CMDP formulation shares this motivation but bounds the expected onward cost of every subprocess rather than the accumulated cost along every prefix. This distinction leads to different algorithmic techniques. Anytime-constrained MDPs motivate state augmentation by cumulative cost or related bookkeeping variables [21]. In contrast, we use value-set reasoning over attainable subprocess values, which enables a direct characterization of feasible continuations and a polynomial-time procedure for computing near-optimal history-dependent policies.

On the algorithmic side, value-set iteration has precedents in game theory, where similar set-valued dynamic programming methods have been used to compute equilibria in sequential games [22, 10, 19, 15, 14]. In these settings, players' incentive constraints are defined in terms of expected future payoffs and must be satisfied in every subgame, mirroring the subprocess constraints in our model. The recent work of Gan and Majumdar [14] provides a detailed algorithmic study of value-set iteration for computing extensive-form correlated equilibria in Markov games. Our work adapts this framework to S-CMDPs, a setting that we believe admits an even broader range of applications in planning, control, and RL. At a technical level, our VSI framework rests on a fixed point of a monotone operator. The computation of Tarski fixed points has recently attracted growing attention in theoretical computer science [11, 12, 9, 6].

B Omitted Proofs

B.1 Proofs in Section 3

Theorem 1. *Fix any discount factor $\gamma \in (0, 1)$. There exist constants $\epsilon, \delta > 0$ such that computing a **stationary** policy that is (ϵ, δ)-optimal among stationary policies is NP-hard.*

Proof. We reduce from the GAP 3-SAT problem. An instance is a 3CNF formula φ, promised to be either satisfiable (a yes-instance) or such that no Boolean assignment satisfies more than a β fraction of its clauses (a no-instance). This problem is NP-hard for any constant $\beta \in (7/8, 1)$ [18].

Consider a GAP 3-SAT instance φ. Let $x_1, \ldots, x_m$ be its Boolean variables and $C_1, \ldots, C_n$ its clauses. Each clause is a disjunction of three literals, each of which is a variable x_i or its negation $\neg x_i$.

We construct an S-CMDP instance in which every trajectory reaches an absorbing terminal state within four steps. The state space is $\{s_0, s_\perp\} \cup \{x_1, \ldots, x_m\} \cup \bigcup_{j=1}^{n} \{C_j, \ell_{j1}, \ell_{j2}, \ell_{j3}\}$. An example of the reduction is illustrated in Figure 2. In more detail:

- $s_\perp$ is an absorbing terminal state with zero reward and cost. Each state x_i corresponds to the variable x_i in the GAP 3-SAT instance; each state C_j corresponds to the clause C_j; and each state ℓ_{jk} corresponds to the k-th literal in C_j.
- The initial state is s_0. Irrespective of the action taken, it transitions to each clause state C_j with probability $1/n$, generating zero reward and cost.

  All other transitions are deterministic:

  - There are four actions available at each C_j: one leading to $s_\perp$, and three others each leading to one of the literal states $\ell_{j1}, \ell_{j2}, \ell_{j3}$. All these transitions generate zero reward and zero cost; for all $a \in A$:

    $$r(C_j, a) = c(C_j, a) = 0.$$

    (The action leading to $s_\perp$ ensures that there is a feasible policy for every instance.)
  - At each literal state ℓ_{jk}, the only available action leads to the corresponding variable state x_i. Its reward and cost depend on the sign of the literal:

    $$r(\ell_{jk}, a) = c(\ell_{jk}, a) = \begin{cases} \gamma & \text{if } \ell_{jk} = x_i, \\ 0 & \text{if } \ell_{jk} = \neg x_i. \end{cases}$$

  - At each variable state x_i, there are two actions available; call them a_0 and a_1. Both actions lead to $s_\perp$, generating the following reward and cost:

    $$\begin{aligned} & r(x_i, a_0) = c(x_i, a_0) = 0, \\ \text{and} \quad & r(x_i, a_1) = c(x_i, a_1) = 1. \end{aligned}$$

  Hence, the reward and cost are equal for every state-action pair in the S-CMDP instance.
- Finally, we set the thresholds in the constraints as follows.

  $$\theta(s) = \begin{cases} \gamma & \text{if } s \text{ is a positive literal state,} \\ 2 & \text{otherwise.} \end{cases}$$

  All onward costs are at most 2, so only the positive literal states have nontrivial constraints.

The construction has polynomial size. State-dependent action sets can be replaced by a common action set by duplicating available actions.

Fix $\beta \in (7/8, 1)$, choose $\eta \in (0, 1 - \beta)$, and set $\delta = \gamma\eta$. We establish the following two claims:

- **Completeness:** If the GAP 3-SAT instance is a yes-instance, then there exists a feasible stationary policy π with $V_r^\pi(s_0) = \gamma^3$.
- **Soundness:** If it is a no-instance, then $V_r^\pi(s_0) \leq (\beta + \eta)\gamma^3$ for every δ-feasible stationary policy π.

Taking $\epsilon = (1 - \beta - \eta)\gamma^3/2 > 0$ then leaves a strict gap between the value guaranteed in a yes-instance and the upper bound in a no-instance. Note that it is essential that, in the completeness condition, the yes-instance corresponds to a policy that satisfies the constraints *exactly* because of the resource augmentation in the approximation criterion considered (Definition 1).

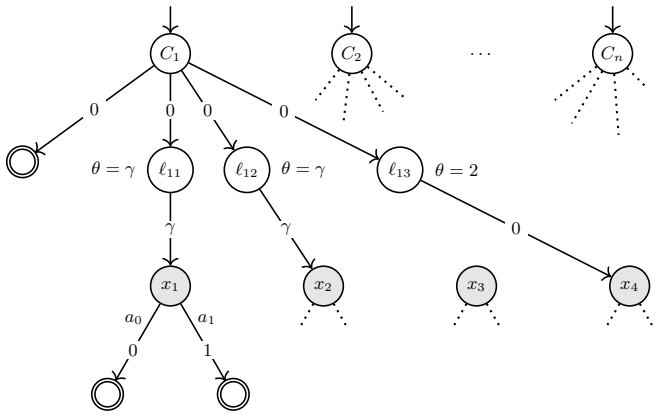


Figure 2: Reduction from GAP 3-SAT. In this example, there are four variables $x_1, \ldots, x_4$ in φ and $C_1 = (x_1 \vee x_2 \vee \neg x_4)$. The states connected to C_1 are shown. The initial state s_0, which transitions uniformly to the clause states, is omitted. Edge labels give the reward, which equals the cost in this reduction.

Proof of Completeness Suppose that φ is a yes-instance, so there exists a satisfying assignment $f : \{x_1, \ldots, x_m\} \to \{true, false\}$. Consider the following stationary policy $\pi : S \to \Delta(A)$. We show that $V_r^\pi(s_0) = \gamma^3$.

- At each variable state x_i, choose a_0 if $f(x_i) = true$, and a_1 otherwise.
- At each clause state C_j, choose with probability 1 an action leading to a literal that is true under f. Such a literal exists because f satisfies φ.

If the chosen literal is positive, its immediate reward and cost are γ, and the next action is a_0, with zero reward and cost. If the chosen literal is negative, its immediate reward and cost are zero, and the next action is a_1, with reward and cost 1. Thus, in both cases the onward reward and cost at the literal state equal γ. All constraints are satisfied, each clause state has reward value γ^2, and $V_r^\pi(s_0) = \gamma^3$.

Proof of Soundness Suppose that φ is a no-instance, and let π be any δ-feasible stationary policy. Write

$$q_i = \pi(a_1 \,|\, x_i).$$

At a reachable positive literal corresponding to x_i, the onward cost is $\gamma + \gamma q_i$. Since its threshold is γ, δ-feasibility requires $q_i \leq \delta/\gamma = \eta$. Thus, a positive literal with $q_i > \eta$ cannot be reached under π.

Let $T = \{x_i \mid q_i \leq \eta\}$ and let F contain the remaining variables. We bound the reward at each clause state C_j by considering three cases:

- **Case 1:** C_j contains a positive literal x_i with $x_i \in T$. Selecting such a literal gives reward $\gamma^2(1 + q_i) \leq (1 + \eta)\gamma^2$ at C_j. Any selectable positive literal satisfies the same bound, and any negative literal gives at most γ^2. Thus $V_r^\pi(C_j) \leq (1 + \eta)\gamma^2$.
- **Case 2:** Case 1 does not apply, but C_j contains a negative literal $\neg x_i$ with $x_i \in F$. All positive literals in the clause are unreachable under π, and selecting any negative literal gives reward $\gamma^2 q_i \leq \gamma^2$ at C_j. Hence $V_r^\pi(C_j) \leq \gamma^2$.
- **Case 3:** Neither of the above cases applies. Every positive literal has its variable in F and is therefore unreachable, while every negative literal has its variable in T and gives reward at most $\eta\gamma^2$ at C_j. The terminal action gives zero reward, so $V_r^\pi(C_j) \leq \eta\gamma^2$.

Assigning $true$ to the variables in T and $false$ to those in F satisfies every clause in Cases 1 and 2. Since φ is a no-instance, these cases cover at most a β fraction of the clauses. Their reward values

are at most $(1+\eta)\gamma^2$, while the remaining clause values are at most $\eta\gamma^2$. Accounting for the initial transition from s_0, we obtain

$$V_r^\pi(s_0) \;=\; \frac{\gamma}{n}\sum_{j=1}^{n} V_r^\pi(C_j) \;\le\; \gamma\big(\beta(1+\eta)\gamma^2 + (1-\beta)\eta\gamma^2\big) \;=\; (\beta+\eta)\gamma^3.$$

In a yes-instance, any (ϵ,δ)-optimal stationary policy has value at least $\gamma^3-\epsilon$. In a no-instance, every δ-feasible stationary policy has value at most $(\beta+\eta)\gamma^3$. Since $\gamma^3-\epsilon > (\beta+\eta)\gamma^3$, computing such a policy would distinguish the two cases, proving the claimed hardness. Since all nonzero rewards and costs occur within four steps, the result also holds for a finite horizon. □

B.2 Proofs in Section 5

Lemma 2. *For any $s \in S$ and $\mathbf{v} = (v_1, v_2) \in \mathbb{R}^2$, we have $\mathbf{v} \in \mathcal{V}^\star(s)$ if and only if $v_2 \le \theta(s)$ and there exist a distribution $\bar\pi \in \Delta(A)$ and a vector $\mathbf{w}(a,s') \in \mathbb{R}^2$ for each $(a,s') \in A \times S$ such that Eqs.* (3) *and* (4) *hold.*

Proof. The "only if" direction follows directly from the Bellman equation and the subprocess feasibility requirement.

For the "if" direction, suppose $v_2 \le \theta(s)$ and there exist $\bar\pi$ and $\mathbf{w}$ satisfying the stated conditions. By Eq. (4), for each pair (a,s') with $\bar\pi(a)\cdot p(s' \mid s,a) > 0$, there exists a feasible policy $\pi_{a,s'}$ that achieves the onward vector $\mathbf{w}(a,s')$ from state s'. We construct a policy π by concatenating $\bar\pi$ and these policies: π selects its initial action at s according to $\bar\pi$, and thereafter follows $\pi_{a,s'}$ once action a is taken and state s' is reached. Its action distributions on unreachable histories can be chosen arbitrarily. Since $v_2 \le \theta(s)$ and each $\pi_{a,s'}$ is feasible, the resulting policy π satisfies the subprocess constraints. Applying the Bellman equation then yields $\vec{V}^\pi(s) = \mathbf{v}$. Hence $\mathbf{v}$ is achievable: $\mathbf{v} \in \mathcal{V}^\star(s)$. □

Lemma 4 (Monotonicity)**.** *If $\mathcal{V} \subseteq \mathcal{V}'$, then $\Phi(\mathcal{V}) \subseteq \Phi(\mathcal{V}')$.*

Proof. It is immediate from the definition that $\mathrm{Onward}(s,\cdot)$ is monotone. Hence, if $\mathcal{V} \subseteq \mathcal{V}'$, then we have $\mathrm{Onward}(s,\mathcal{V}) \subseteq \mathrm{Onward}(s,\mathcal{V}')$. According to the definition of Φ, enlarging the feasible set $\mathrm{Onward}(s,\mathcal{V})$ can only enlarge the resulting value set. Therefore, $\Phi(\mathcal{V})(s) \subseteq \Phi(\mathcal{V}')(s)$ for all $s \in S$. □

Lemma 5. *$\mathcal{V}^\star \subseteq \mathcal{B}$ and $\Phi(\mathcal{B}) \subseteq \mathcal{B}$.*

Proof. The inclusion $\mathcal{V}^\star \subseteq \mathcal{B}$ follows from the boundedness of discounted returns. Since rewards and costs are assumed to lie in $[-1,1]$, the discounted cumulative return of any policy is bounded in magnitude by the geometric sum $\sum_{t=0}^{\infty}\gamma^t = 1/(1-\gamma)$. Hence every achievable value vector lies in $\mathcal{B}$.

For the second inclusion, consider any $\mathbf{v} \in \Phi(\mathcal{B})(s)$. By definition of Φ, $\mathbf{v}$ is obtained from an immediate payoff of magnitude at most 1 together with an onward value bounded by $1/(1-\gamma)$. Therefore

$$\|\mathbf{v}\|_\infty \;\le\; 1 + \gamma\cdot\frac{1}{1-\gamma} \;=\; \frac{1}{1-\gamma},$$

which implies $\mathbf{v} \in \mathcal{B}$. Hence $\Phi(\mathcal{B}) \subseteq \mathcal{B}$. □

Lemma 6. *$\mathcal{V}^\star \subseteq \mathcal{V}^{k+1} \subseteq \mathcal{V}^k$ for all $k \ge 0$.*

Proof. We first prove the base case $\mathcal{V}^\star \subseteq \mathcal{V}^1 \subseteq \mathcal{V}^0$.

Since $\mathcal{V}^\star \subseteq \mathcal{B}$ (Lemma 5), monotonicity of Φ implies

$$\Phi(\mathcal{V}^\star) \subseteq \Phi(\mathcal{B}).$$

Recall that $\mathcal{V}^\star = \Phi(\mathcal{V}^\star)$ (Corollary 3). Moreover, by definition of the sequence $(\mathcal{V}^k)_k$, we have $\mathcal{V}^0 = \mathcal{B}$ and $\mathcal{V}^1 = \Phi(\mathcal{V}^0)$. Combining these results establishes

$$\mathcal{V}^\star = \Phi(\mathcal{V}^\star) \subseteq \Phi(\mathcal{B}) = \mathcal{V}^1.$$

Together with $\Phi(\mathcal{B}) \subseteq \mathcal{B}$ (Lemma 5), this yields the base case $\mathcal{V}^\star \subseteq \mathcal{V}^1 \subseteq \mathcal{V}^0$.

Now suppose that $\mathcal{V}^\star \subseteq \mathcal{V}^{k+1} \subseteq \mathcal{V}^k$ holds for some $k \geq 0$. Applying Φ to the inclusions and using monotonicity gives

$$\Phi(\mathcal{V}^\star) \subseteq \Phi(\mathcal{V}^{k+1}) \subseteq \Phi(\mathcal{V}^k).$$

By definition of the sequence $(\mathcal{V}^k)_k$ and the fact that $\mathcal{V}^\star = \Phi(\mathcal{V}^\star)$, this becomes $\mathcal{V}^\star \subseteq \mathcal{V}^{k+2} \subseteq \mathcal{V}^{k+1}$.

By induction, the stated result then follows. □

Lemma 7. *If $\mathcal{V}$ is bounded and $\mathcal{V} = \Phi(\mathcal{V})$, then $\mathcal{V} \subseteq \mathcal{V}^\star$.*

Proof. Suppose $\mathcal{V}$ is bounded and $\mathcal{V} = \Phi(\mathcal{V})$. Fix $s \in S$ and $\mathbf{v} \in \mathcal{V}(s)$. We construct a policy π and show that it is feasible and satisfies $\vec{V}^\pi(s) = \mathbf{v}$.

Recursive Policy Construction For a history h, let s_h denote its last state. We define the policy π and targets $\mathbf{v}_h \in \mathcal{V}(s_h)$ recursively over reachable histories:

- At the initial history $h_0 = (s)$, assign the target $\mathbf{v}_{h_0} = \mathbf{v}$.
- At each reachable history h with target $\mathbf{v}_h$, choose $(\bar{\pi}_h, \mathbf{w}_h) \in \mathrm{Onward}(s_h, \mathcal{V})$ such that

  $$\mathrm{Bellman}(s_h, \bar{\pi}_h, \mathbf{w}_h) = \mathbf{v}_h.$$

  Such a pair exists because $\mathcal{V} = \Phi(\mathcal{V})$. Set

  $$\pi(h) = \bar{\pi}_h.$$

- For every pair (a, s') with $\bar{\pi}_h(a) \cdot p(s' \mid s_h, a) > 0$, assign to the reachable extension $h' = (h, a, s')$ the target

  $$\mathbf{v}_{h'} = \mathbf{w}_h(a, s').$$

  Crucially, the onward condition imposed on $\mathbf{w}_h$ ensures that $\mathbf{v}_{h'} \in \mathcal{V}(s')$, so the construction can continue at h'.

For completeness, for all unreachable histories h, we assign an arbitrary action distribution to $\pi(h)$.

Exact Realization Let $\vec{V}^\pi(h)$ denote the true onward value of π conditional on h. Its Bellman expansion and the exact decomposition of $\mathbf{v}_h$ use the same immediate reward and cost terms. Subtracting them and using the successor targets gives

$$\vec{V}^\pi(h) - \mathbf{v}_h = \gamma \, \mathbb{E}_\pi \left[\vec{V}^\pi(h') - \mathbf{v}_{h'} \,\middle|\, h \right],$$
$$\|\vec{V}^\pi(h) - \mathbf{v}_h\|_\infty \leq \gamma \, \mathbb{E}_\pi \left[\|\vec{V}^\pi(h') - \mathbf{v}_{h'}\|_\infty \,\middle|\, h \right],$$

where h' is the next history obtained from h under π and p, and the inequality follows from the triangle inequality. Since $\mathcal{V}$ is bounded, let $M < \infty$ bound the L_∞ norms of its vectors. Every target then has norm at most M, while every true onward value has norm at most $1/(1-\gamma)$. Iterating the inequality for $T \geq 1$ steps therefore yields

$$\|\vec{V}^\pi(h) - \mathbf{v}_h\|_\infty \leq \gamma^T \left(\frac{1}{1-\gamma} + M \right).$$

Letting $T \to \infty$ gives $\vec{V}^\pi(h) = \mathbf{v}_h$ at every reachable history. In particular, $\vec{V}^\pi(s) = \mathbf{v}$.

Feasibility At every reachable history h, we have $\mathbf{v}_h \in \mathcal{V}(s_h) = \Phi(\mathcal{V})(s_h)$, so $v_{h,2} \le \theta(s_h)$. Since $\vec{V}^\pi(h) = \mathbf{v}_h$, the policy π is feasible. Thus $\mathbf{v} \in \mathcal{V}^\star(s)$, proving $\mathcal{V} \subseteq \mathcal{V}^\star$. □

Theorem 8. *$\mathcal{V}^k \to \mathcal{V}^\star$ as $k \to \infty$.*

Proof. Let $\mathcal{V}^\infty(s) := \bigcap_{k\ge 0} \mathcal{V}^k(s)$ for every $s \in S$. By Lemma 6, we already have $\mathcal{V}^\star \subseteq \mathcal{V}^\infty \subseteq \mathcal{B}$. Thus, by Lemma 7, it suffices to show that $\mathcal{V}^\infty$ is a fixed point of Φ. We prove the two inclusions separately.

The Inclusion $\mathcal{V}^\infty \subseteq \Phi(\mathcal{V}^\infty)$ We construct a Bellman decomposition in $\mathcal{V}^\infty$ by taking a limit of decompositions in the iterates. Fix an arbitrary state $s \in S$ and vector $\mathbf{v} \in \mathcal{V}^\infty(s)$. By definition, $\mathbf{v} \in \mathcal{V}^k(s)$ for all $k \ge 0$. Since $\mathcal{V}^{k+1} = \Phi(\mathcal{V}^k)$, we also have $\mathbf{v} \in \Phi(\mathcal{V}^k)(s)$. According to the definition of Φ, for each k, there exists $(\bar\pi^k, \mathbf{w}^k) \in \mathrm{Onward}(s, \mathcal{V}^k)$ such that

$$\mathbf{v} = \mathrm{Bellman}(s, \bar\pi^k, \mathbf{w}^k).$$

Choose $\mathbf{w}^k(a, s') = \mathbf{0}$ whenever $\bar\pi^k(a) \cdot p(s' \mid s, a) = 0$; this does not affect the Bellman expansion. All onward vectors then lie in $\mathcal{B}$. Consider the sequence $\left(\bar\pi^k, \mathbf{w}^k\right)_{k\ge 0}$. Since both components are bounded, by the Bolzano–Weierstrass theorem, this sequence admits a convergent subsequence $\left(\bar\pi^{k_\ell}, \mathbf{w}^{k_\ell}\right)_{\ell\ge 0}$. Let $(\bar\pi, \mathbf{w})$ be the limit of this subsequence.

The key is to show that the limiting onward vectors for positive-probability pairs remain in every iterate, and hence in their intersection.

Claim 1. $(\bar\pi, \mathbf{w}) \in \mathrm{Onward}(s, \mathcal{V}^\infty)$.

Proof. According to the definition of $\mathrm{Onward}(s, \cdot)$ in Eq. (7), we need to show that:

(i) $\bar\pi \in \Delta(A)$; and

(ii) $\mathbf{w}(a, s') \in \mathcal{V}^\infty(s')$ for all a and s' such that $\bar\pi(a) \cdot p(s' \mid s, a) > 0$.

For (i), note that $\Delta(A)$ is closed and each $\bar\pi^{k_\ell} \in \Delta(A)$. Since $\bar\pi$ is their limit, it follows that $\bar\pi \in \Delta(A)$.

For (ii), fix $a \in A$ and $s' \in S$ such that $\bar\pi(a) \cdot p(s' \mid s, a) > 0$. Let $k \ge 0$ be arbitrary. Since $k_\ell \to \infty$ and $\bar\pi^{k_\ell}(a) \to \bar\pi(a) > 0$, there exists L such that $k_\ell \ge k$ and $\bar\pi^{k_\ell}(a) \cdot p(s' \mid s, a) > 0$ for all $\ell \ge L$. Because $(\bar\pi^{k_\ell}, \mathbf{w}^{k_\ell}) \in \mathrm{Onward}(s, \mathcal{V}^{k_\ell})$ we have

$$\mathbf{w}^{k_\ell}(a, s') \;\in\; \mathcal{V}^{k_\ell}(s') \;\subseteq\; \mathcal{V}^k(s')$$

for all $\ell \ge L$.

To pass to the limit, we use the fact that Φ preserves closedness of bounded closed value sets. Indeed, all onward vectors can be restricted to a closed ball containing the input sets. The onward constraints remain valid under limits: any action–state pair with positive limiting probability has positive probability for all sufficiently large indices, so its onward vectors lie in the corresponding closed input set, which also contains their limit. Together with the closed cost constraint, this makes the feasible decompositions a compact set. Its continuous Bellman image is compact and hence closed. Since $\mathcal{V}^0(s') = \mathcal{B}$ is closed and all iterates are bounded, induction shows that every $\mathcal{V}^k(s')$ is closed.

Taking limits therefore gives $\mathbf{w}(a, s') \in \mathcal{V}^k(s')$. Since k is arbitrary, we conclude that $\mathbf{w}(a, s') \in \bigcap_{k\ge 0} \mathcal{V}^k(s') = \mathcal{V}^\infty(s')$. □

We can now pass to the limit in the Bellman expansion. Since $\mathbf{v} = \mathrm{Bellman}(s, \bar\pi^{k_\ell}, \mathbf{w}^{k_\ell})$ for all ℓ, and $\mathrm{Bellman}(s, \cdot, \cdot)$ is continuous, taking limits along the subsequence yields $\mathbf{v} = \mathrm{Bellman}(s, \bar\pi, \mathbf{w})$. Moreover, $v_2 \le \theta(s)$ since $\mathbf{v} \in \mathcal{V}^1(s)$. Together with Claim 1, this implies $\mathbf{v} \in \Phi(\mathcal{V}^\infty)(s)$. Because the choice of $s \in S$ and $\mathbf{v} \in \mathcal{V}^\infty(s)$ is arbitrary, we conclude that

$$\mathcal{V}^\infty \subseteq \Phi(\mathcal{V}^\infty).$$

The Reverse Inclusion Since $\mathcal{V}^\infty \subseteq \mathcal{V}^k$ for every $k \geq 0$, monotonicity gives $\Phi(\mathcal{V}^\infty) \subseteq \Phi(\mathcal{V}^k) = \mathcal{V}^{k+1}$. Taking intersections yields, for every $s \in S$,

$$\Phi(\mathcal{V}^\infty)(s) \subseteq \bigcap_{k\geq 0} \mathcal{V}^{k+1}(s) = \bigcap_{k\geq 0} \mathcal{V}^k(s) = \mathcal{V}^\infty(s),$$

where dropping $\mathcal{V}^0(s)$ does not change the intersection because the iterates are nested.

Therefore, $\mathcal{V}^\infty = \Phi(\mathcal{V}^\infty)$. By Lemma 7, $\mathcal{V}^\infty \subseteq \mathcal{V}^\star$, and the reverse inclusion from Lemma 6 gives $\mathcal{V}^\infty = \mathcal{V}^\star$. □

B.3 Proofs in Section 6

Lemma 9. *$\mathcal{V}^\star \subseteq \widehat{\mathcal{V}}^{k+1} \subseteq \widehat{\mathcal{V}}^k$ for all $k \geq 0$.*

Proof. Let r_ξ denote the radius of $\widehat{\mathcal{B}}_\xi$. By definition,

$$r_\xi > \frac{1}{1-\gamma} \qquad \text{and} \qquad r_\xi \in \xi\mathbb{N},$$

and $\widehat{\mathcal{V}}^0 = \widehat{\mathcal{B}}_\xi$.

We first establish (i) $\mathcal{V}^\star \subseteq \widehat{\mathcal{V}}^0$, and (ii) $\widehat{\Phi}(\widehat{\mathcal{V}}^0) \subseteq \widehat{\mathcal{V}}^0$. Indeed, condition (i) is immediate, since

$$\mathcal{V}^\star \subseteq \mathcal{B} \subseteq \widehat{\mathcal{B}}_\xi = \widehat{\mathcal{V}}^0.$$

To verify (ii), fix any $s \in S$ and $\mathbf{v} \in \Phi(\widehat{\mathcal{V}}^0)(s)$. Since the immediate return is bounded in L_∞ norm by 1 and every onward value in $\widehat{\mathcal{V}}^0$ has L_∞ norm at most r_ξ, we get that

$$\|\mathbf{v}\|_\infty \leq 1 + \gamma r_\xi < r_\xi,$$

where the strict inequality follows from $r_\xi > 1/(1-\gamma)$.

Now consider any

$$\mathbf{g} \in \mathrm{neigh}_\xi\Big(\Phi(\widehat{\mathcal{V}}^0)(s)\Big) \cap G_\xi.$$

There exists $\mathbf{v} \in \Phi(\widehat{\mathcal{V}}^0)(s)$ such that $\|\mathbf{g} - \mathbf{v}\|_\infty \leq \xi$, and hence $\|\mathbf{g}\|_\infty < r_\xi + \xi$. Since $\mathbf{g} \in G_\xi$, $\|\mathbf{g}\|_\infty$ is a multiple of ξ. As r_ξ is also a multiple of ξ, it follows that

$$\|\mathbf{g}\|_\infty \leq r_\xi.$$

Thus every such grid point belongs to $\widehat{\mathcal{V}}^0$. Since $\widehat{\mathcal{V}}^0$ is convex, by the definition of $\widehat{\Phi}$, we get that

$$\widehat{\Phi}(\widehat{\mathcal{V}}^0)(s) \;=\; \mathrm{GridHull}_\xi\Big(\mathrm{neigh}_\xi\Big(\Phi(\widehat{\mathcal{V}}^0)(s)\Big)\Big) \;\subseteq\; \widehat{\mathcal{V}}^0.$$

Since s was arbitrary, condition (ii) follows.

Finally, observe that for any value-set function $\mathcal{V}$, we have

$$\Phi(\mathcal{V})(s) \subseteq \widehat{\Phi}(\mathcal{V})(s) \qquad \text{for every } s \in S.$$

Indeed, fix any $\mathbf{v} \in \Phi(\mathcal{V})(s)$ and consider a grid cell containing $\mathbf{v}$. Every vertex $\mathbf{g}$ of this cell belongs to G_ξ and satisfies $\|\mathbf{g} - \mathbf{v}\|_\infty \leq \xi$, so

$$\mathbf{g} \in \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s)) \cap G_\xi.$$

Since $\mathbf{v}$ is a convex combination of the vertices of the cell, the definition of $\mathrm{GridHull}_\xi$ then implies $\mathbf{v} \in \widehat{\Phi}(\mathcal{V})(s)$; therefore, $\Phi(\mathcal{V})(s) \subseteq \widehat{\Phi}(\mathcal{V})(s)$.

Applying this to $\mathcal{V}^\star$ and using $\mathcal{V}^\star = \Phi(\mathcal{V}^\star)$ gives

$$\mathcal{V}^\star = \Phi(\mathcal{V}^\star) \subseteq \widehat{\Phi}(\mathcal{V}^\star).$$

Together with the monotonicity of $\widehat{\Phi}$, conditions (i) and (ii) give

$$\mathcal{V}^\star \subseteq \widehat{\mathcal{V}}^1 \subseteq \widehat{\mathcal{V}}^0.$$

Applying $\widehat{\Phi}$ repeatedly and using $\mathcal{V}^\star \subseteq \widehat{\Phi}(\mathcal{V}^\star)$ then yields

$$\mathcal{V}^\star \subseteq \widehat{\mathcal{V}}^{k+1} \subseteq \widehat{\mathcal{V}}^k$$

for every $k \geq 0$. □

Lemma 10. *If $\mathcal{V}$ is bounded and $\mathcal{V} = \widehat{\Phi}(\mathcal{V})$, then for all $s \in S$ every $\mathbf{v} \in \mathcal{V}(s)$ is (ϵ, δ)-achievable from s.*

Proof. Suppose $\mathcal{V}$ is bounded and $\mathcal{V} = \widehat{\Phi}(\mathcal{V})$. Fix $s \in S$ and $\mathbf{v} \in \mathcal{V}(s)$. We construct a policy π and show that it is δ-feasible and satisfies $\|\vec{V}^{\pi}(s) - \mathbf{v}\|_{\infty} \leq \epsilon$. The construction uses a relaxed version of `Achieve` (Alg. 2), allowing a Bellman error of at most ξ at each step.

Recursive Policy Construction We first note that, for every $s' \in S$,

$$\widehat{\Phi}(\mathcal{V})(s') = \mathrm{GridHull}_{\xi}\big(\mathrm{neigh}_{\xi}(\Phi(\mathcal{V})(s'))\big) \subseteq \mathrm{neigh}_{\xi}(\Phi(\mathcal{V})(s'))\,.$$

Indeed, $\mathcal{V} = \widehat{\Phi}(\mathcal{V})$ has convex values, so $\Phi(\mathcal{V})(s')$ is convex by mixing Bellman decompositions. Its ξ-neighborhood is therefore convex and contains the convex hull of its grid points. Using $\mathcal{V} = \widehat{\Phi}(\mathcal{V})$ now gives

$$\mathcal{V}(s') \subseteq \mathrm{neigh}_{\xi}(\Phi(\mathcal{V})(s'))\,.$$

For a history h, let s_h denote its last state. We define the policy π and targets $\mathbf{v}_h \in \mathcal{V}(s_h)$ recursively over reachable histories:

- At the initial history $h_0 = (s)$, assign the target $\mathbf{v}_{h_0} = \mathbf{v}$.
- At each reachable history h with target $\mathbf{v}_h$, choose $(\bar{\pi}_h, \mathbf{w}_h) \in \mathrm{Onward}(s_h, \mathcal{V})$ such that

  $$\mathrm{Bellman}(s_h, \bar{\pi}_h, \mathbf{w}_h) \in \mathrm{neigh}_{\xi}(\mathbf{v}_h), \tag{13}$$

  and set

  $$\pi(h) = \bar{\pi}_h.$$

  Such a choice exists because the inclusion above ensures that some vector in $\Phi(\mathcal{V})(s_h)$ lies within distance ξ of $\mathbf{v}_h$. By the definition of Φ, that vector admits a Bellman decomposition using a pair in $\mathrm{Onward}(s_h, \mathcal{V})$, which satisfies Eq. (13).
- For every pair (a, s') with $\bar{\pi}_h(a) \cdot p(s' \mid s_h, a) > 0$, assign to the reachable extension $h' = (h, a, s')$ the target

  $$\mathbf{v}_{h'} = \mathbf{w}_h(a, s') \in \mathcal{V}(s').$$

  The membership $\mathbf{w}_h(a, s') \in \mathcal{V}(s')$ follows from the onward condition; it ensures that the recursive construction can continue at h'.

Define π arbitrarily on unreachable histories. The construction uses precisely the relaxed expansion in Alg. 5, with $\mathcal{V}$ in place of $\widehat{\mathcal{V}}^{\star}$.

Approximation Error For each reachable history h, write $\tilde{\mathbf{v}}_h = \mathrm{Bellman}(s_h, \bar{\pi}_h, \mathbf{w}_h)$. By Eq. (13),

$$\|\tilde{\mathbf{v}}_h - \mathbf{v}_h\|_{\infty} \leq \xi. \tag{14}$$

Let $\vec{V}^{\pi}(h)$ denote the true onward value of π conditional on h. Define the onward-value function $\vec{W}_h$ by

$$\vec{W}_h(a, s') = \vec{V}^{\pi}(h, a, s')$$

for each reachable extension (h, a, s'), assigning arbitrary values on zero-probability pairs. The Bellman equation then gives

$$\vec{V}^{\pi}(h) = \mathrm{Bellman}(s_h, \bar{\pi}_h, \vec{W}_h), \qquad \tilde{\mathbf{v}}_h = \mathrm{Bellman}(s_h, \bar{\pi}_h, \mathbf{w}_h).$$

Both expressions use the same immediate reward and cost terms. Subtracting them and using the successor targets gives

$$\vec{V}^{\pi}(h) - \tilde{\mathbf{v}}_h = \gamma\, \mathbb{E}_{\pi}\Big[\vec{V}^{\pi}(h') - \mathbf{v}_{h'} \,\Big|\, h\Big]\,,$$

where h' is the next history obtained from h under π and p. Taking the L_∞ norm and applying the triangle inequality and Eq. (14) gives

$$\begin{aligned}\|\vec{V}^\pi(h) - \mathbf{v}_h\|_\infty &\le \|\vec{V}^\pi(h) - \tilde{\mathbf{v}}_h\|_\infty + \|\tilde{\mathbf{v}}_h - \mathbf{v}_h\|_\infty \\ &\le \xi + \gamma\, \mathbb{E}_\pi\Big[\|\vec{V}^\pi(h') - \mathbf{v}_{h'}\|_\infty \,\Big|\, h\Big] \\ &\le \xi + \gamma \max_{h'} \|\vec{V}^\pi(h') - \mathbf{v}_{h'}\|_\infty.\end{aligned}$$

Here, the expectation is a weighted average over all histories h' reachable from h in one step, so it is bounded by the maximum over those histories. Since $\mathcal{V}$ is bounded, let $M < \infty$ bound the L_∞ norms of its vectors. Every target then has norm at most M, while every true onward value has norm at most $1/(1-\gamma)$. Iterating this inequality for $T \ge 1$ steps therefore yields

$$\|\vec{V}^\pi(h) - \mathbf{v}_h\|_\infty \le \xi \sum_{j=0}^{T-1} \gamma^j + \gamma^T \left(\frac{1}{1-\gamma} + M\right).$$

Letting $T \to \infty$ gives

$$\|\vec{V}^\pi(h) - \mathbf{v}_h\|_\infty \le \frac{\xi}{1-\gamma} \le \epsilon. \tag{15}$$

Applying this at $h_0 = (s)$ gives $\|\vec{V}^\pi(s) - \mathbf{v}\|_\infty \le \epsilon$.

Feasibility Every target $\mathbf{v}_h$ lies within distance ξ of $\Phi(\mathcal{V})(s_h)$, whose vectors satisfy the cost threshold at s_h. Hence

$$v_{h,2} \le \theta(s_h) + \xi. \tag{16}$$

Together with Eq. (15), this yields, at every reachable history h,

$$V_2^\pi(h) \le v_{h,2} + \frac{\xi}{1-\gamma} \le \theta(s_h) + \xi + \frac{\xi}{1-\gamma} \le \theta(s_h) + \delta,$$

where the final inequality follows from $\xi \le (1-\gamma)\min\{\epsilon, \delta/2\}$. Thus π is δ-feasible and $\mathbf{v}$ is (ϵ, δ)-achievable from s. □

Lemma 11. *Suppose each $\mathcal{V}(s')$ is given as the convex hull of finitely many grid points. For any s and $\mathbf{v}$, the feasibility of Eqs.* (9) *to* (12) *can be decided by solving an LP.*

Proof. The constraints involve two difficulties. First, Eq. (10) involves quadratic terms of the form $\bar{\pi}(a) \cdot \mathbf{w}(a, s')$. Second, the index selection condition in Eq. (12) depends on the variable $\bar{\pi}(a)$. We show that both issues can be eliminated to obtain an equivalent formulation with only linear constraints.

Removing the Index Condition Observe that Eq. (12) is equivalent to imposing the following conditions for every $a \in A$ and $s' \in S$, where we write $\bar{\pi}(a) \cdot \mathcal{V}(s') = \{\bar{\pi}(a) \cdot \mathbf{v} \mid \mathbf{v} \in \mathcal{V}(s')\}$:

$$\begin{aligned}&\bar{\pi}(a) \cdot \mathbf{w}(a, s') \in \bar{\pi}(a) \cdot \mathcal{V}(s') && \text{if } p(s' \mid s, a) > 0 \text{ and } \mathcal{V}(s') \neq \varnothing, \\ &\bar{\pi}(a) = 0 && \text{if } p(s' \mid s, a) > 0 \text{ and } \mathcal{V}(s') = \varnothing.\end{aligned}$$

The first constraint holds automatically if $\bar{\pi}(a) = 0$; and if $\bar{\pi}(a) > 0$, it reduces to $\mathbf{w}(a, s') \in \mathcal{V}(s')$. Thus the dependence of the index condition on $\bar{\pi}(a)$ is removed.

Linearizing Quadratic Terms Let $G(s')$ be the finite set of grid points representing $\mathcal{V}(s')$, so that $\mathcal{V}(s') = \mathrm{conv}(G(s'))$. We introduce convex-weight variables only for pairs (a, s') with $p(s' \mid s, a) > 0$ and $G(s') \neq \varnothing$; empty successor sets with positive transition probability are handled by the zero-action-probability condition above. Then any $\bar{\pi}(a) \cdot \mathbf{w}(a, s') \in \bar{\pi}(a) \cdot \mathcal{V}(s')$ can be written as a convex combination

$$\bar{\pi}(a) \cdot \mathbf{w}(a, s') = \sum_{\mathbf{g} \in G(s')} y_{a,s'}(\mathbf{g}) \cdot \big(\bar{\pi}(a) \cdot \mathbf{g}\big),$$

where $\mathbf{y}_{a,s'} = (y_{a,s'}(\mathbf{g}))_{\mathbf{g} \in G(s')}$ satisfies

$$\mathbf{y}_{a,s'} \in \Delta(G(s')). \tag{17}$$

Namely, $\bar{\pi}(a) \cdot \mathcal{V}(s')$ is the convex hull of the grid points scaled by $\bar{\pi}(a)$.

Substituting this representation of $\bar{\pi}(a) \cdot \mathbf{w}(a, s')$ into Eq. (10) and the reformulation of Eq. (12) above, we convert the quadratic terms to $\bar{\pi}(a) \cdot y_{a,s'}(\mathbf{g})$. To eliminate these terms, introduce auxiliary variables

$$z_{a,s'}(\mathbf{g}) = \bar{\pi}(a) \cdot y_{a,s'}(\mathbf{g}).$$

Since $\mathbf{y}_{a,s'} \in \Delta(G(s'))$, the variables $\mathbf{z}_{a,s'} = (z_{a,s'}(\mathbf{g}))_{\mathbf{g} \in G(s')}$ satisfy the following constraints:

$$z_{a,s'}(\mathbf{g}) \geq 0, \qquad \sum_{\mathbf{g} \in G(s')} z_{a,s'}(\mathbf{g}) = \bar{\pi}(a). \tag{18}$$

We replace Eq. (17) with them.

This substitution replaces every product $\bar{\pi}(a) \cdot y_{a,s'}(\mathbf{g})$ with a linear variable. The resulting constraints are linear in $\bar{\pi}$ and the auxiliary variables $\mathbf{z}_{a,s'}$.

Equivalence of the Reformulation The reformulation is trivially equivalent before introducing $\mathbf{z}$. To see that it remains equivalent after we replace $\mathbf{y}$ with $\mathbf{z}$, observe that if $(\tilde{\mathbf{v}}, \bar{\pi}, \mathbf{y})$ satisfies the original formulation, then $(\tilde{\mathbf{v}}, \bar{\pi}, \mathbf{z})$ with $\mathbf{z}_{a,s'} = \bar{\pi}(a)\mathbf{y}_{a,s'}$ satisfies the reformulated constraints.

Conversely, given feasible $(\tilde{\mathbf{v}}, \bar{\pi}, \mathbf{z})$, define, for each pair with $p(s' \mid s, a) > 0$ and $G(s') \neq \varnothing$,

$$\mathbf{y}_{a,s'} = \begin{cases} \mathbf{z}_{a,s'}/\bar{\pi}(a) & \text{if } \bar{\pi}(a) > 0, \\ \text{an arbitrary vector in } \Delta(G(s')) & \text{if } \bar{\pi}(a) = 0. \end{cases}$$

Then $(\tilde{\mathbf{v}}, \bar{\pi}, \mathbf{y})$ satisfies the original formulation. In particular, in the case where $\bar{\pi}(a) = 0$, Eq. (18) implies that $\mathbf{z}_{a,s'} = \mathbf{0}$; hence, any arbitrary $\mathbf{y}_{a,s'} \in \Delta(G(s'))$ gives $\mathbf{z}_{a,s'} = \bar{\pi}(a) \cdot \mathbf{y}_{a,s'}$.

The Full LP Reformulation In summary, the full LP reformulation is as follows. The variables are:

- $\tilde{\mathbf{v}} = (\tilde{v}_1, \tilde{v}_2)$;
- $\bar{\pi}(a)$ for each $a \in A$; and
- $z_{a,s'}(\mathbf{g})$ for each $a \in A$, $s' \in S$, and $\mathbf{g} \in G(s')$.

For uniform notation, we retain the variables for pairs with $p(s' \mid s, a) = 0$; they are unconstrained and contribute nothing to the Bellman expression. The constraints are:

- Eqs. (9) and (11): $\|\mathbf{v} - \tilde{\mathbf{v}}\|_\infty \leq \xi$, and $\tilde{v}_2 \leq \theta(s)$.
- The linearized form of Eq. (10), as follows:

$$\tilde{\mathbf{v}} \;=\; \sum_{a \in A} \bar{\pi}(a) \cdot \mathbf{x}(s, a) \;+\; \gamma \sum_{a \in A} \sum_{s' \in S} p(s' \mid s, a) \sum_{\mathbf{g} \in G(s')} z_{a,s'}(\mathbf{g}) \cdot \mathbf{g}$$

- The linearized form of Eq. (12), for each $a \in A$ and $s' \in S$ such that $p(s' \mid s, a) > 0$:
  - If $\mathcal{V}(s') \neq \varnothing$, then $z_{a,s'}(\mathbf{g}) \geq 0$ for all $\mathbf{g} \in G(s')$ and $\sum_{\mathbf{g} \in G(s')} z_{a,s'}(\mathbf{g}) = \bar{\pi}(a)$;
  - If $\mathcal{V}(s') = \varnothing$, then $\bar{\pi}(a) = 0$.
- Finally, $\bar{\pi} \in \Delta(A)$ so that $\bar{\pi}$ is a valid distribution.

Any feasible $(\tilde{\mathbf{v}}, \bar{\pi}, \mathbf{z})$ yields a feasible solution $(\tilde{\mathbf{v}}, \bar{\pi}, \mathbf{w})$ of the original constraints by setting

$$\mathbf{w}(a, s') = \begin{cases} \sum_{\mathbf{g} \in G(s')} z_{a,s'}(\mathbf{g}) \cdot \mathbf{g}/\bar{\pi}(a) & \text{if } \bar{\pi}(a) > 0, \\ \text{arbitrary} & \text{if } \bar{\pi}(a) = 0. \end{cases}$$

This completes the proof. □

Theorem 12. *For fixed $\gamma \in (0,1)$, choosing $\xi = (1-\gamma)\min\{\epsilon, \delta/2\}$ allows us to compute a fixed point $\widehat{\mathcal{V}}^\star \supseteq \mathcal{V}^\star$ of $\widehat{\Phi}$ in time polynomial in the input size, $1/\epsilon$, and $1/\delta$. If $\widehat{\mathcal{V}}^\star(s_0) \neq \varnothing$, there exists an (ϵ, δ)-optimal policy π whose action distribution $\pi(h)$ can be computed in time polynomial in the same quantities and $|h|$ for any given history h.*

Proof. Let $N_\xi = |G_\xi \cap \widehat{\mathcal{B}}_\xi|$. Each application of $\widehat{\Phi}$ requires at most $|S|N_\xi$ LP feasibility checks: for each state s and grid point $\mathbf{g} \in G_\xi \cap \widehat{\mathcal{B}}_\xi$, we test whether

$$\mathbf{g} \in \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s)),$$

using Lemma 11. The resulting sets can be represented by all accepted grid points, so each LP has polynomial size in the input and N_ξ. By Lemma 9, the sequence is decreasing, and every strict change removes at least one grid point from a state value set. Thus the procedure reaches a fixed point $\widehat{\mathcal{V}}^\star$ within $O(|S|N_\xi)$ iterations. With $\xi = (1-\gamma)\min\{\epsilon, \delta/2\}$ and fixed γ, N_ξ is polynomial in $1/\epsilon$ and $1/\delta$, establishing the first claim.

If $\widehat{\mathcal{V}}^\star(s_0) = \varnothing$, then $\mathcal{V}^\star(s_0) = \varnothing$ by Lemma 9, so no exactly feasible policy exists. Otherwise, select a target at the initial state by solving

$$\mathbf{v}^\star \in \arg\max_{\mathbf{v} \in \widehat{\mathcal{V}}^\star(s_0)} v_1.$$

Since $\widehat{\mathcal{V}}^\star(s_0)$ is a convex polytope described as the convex hull of finitely many grid points (by construction of $\widehat{\Phi}$), the above maximization reduces to a linear program over convex combination weights, and can therefore be solved in polynomial time.

By Lemma 10, the vector $\mathbf{v}^\star$ is (ϵ, δ)-achievable from s_0. Let π be the policy constructed in the proof of Lemma 10, using a fixed deterministic rule for selecting a basic feasible LP solution at each expansion. Since $\mathcal{V}^\star \subseteq \widehat{\mathcal{V}}^\star$, the reward component of $\mathbf{v}^\star$ is at least that of every exactly feasible policy. Hence π is (ϵ, δ)-optimal. We compute its action distributions using $\texttt{ActProbs}_\xi$, which replaces `Expand` in `ActProbs` (Alg. 3) with $\texttt{Expand}_\xi$ (Alg. 5).

The procedure takes $(\mathbf{v}^\star, h)$ as input and returns the action distribution $\pi(h)$ prescribed at history h. At each recursive step on a reachable history, the current target satisfies $\mathbf{v} \in \widehat{\mathcal{V}}^\star(s)$, so a valid output of $\texttt{Expand}_\xi(\mathbf{v}, s)$ can be found by solving an LP of polynomial size.

To bound the bit lengths through these recursive calls, write the target coordinates over a common denominator, using at most L bits for each numerator and the denominator. The LP coefficient matrix is fixed by the instance and grid, and its right-hand side depends affinely on the target. By Cramer's rule, the variables of a basic feasible solution in standard form can therefore be written over a common denominator using at most $L + C$ bits per numerator and denominator, where C is polynomial in the input size and N_ξ. This common denominator cancels when recovering $\mathbf{w}$ by dividing a weighted sum of the z-variables by $\bar{\pi}(a)$. Since the grid coordinates are fixed, the next target also requires at most $L + C'$ bits per numerator and common denominator, for a polynomial bound C' independent of the current target. Thus each expansion adds only polynomially many bits, so both target and LP solution bit lengths remain polynomial after at most $|h|$ expansions.

Using a fixed deterministic polynomial-time LP algorithm to select basic feasible solutions, the procedure performs at most one expansion per state in the input history and returns an arbitrary fixed distribution if the history is unreachable. Its total running time is thus polynomial in the stated quantities and $|h|$.

This yields an (ϵ, δ)-optimal policy π whose action distribution $\pi(h)$ can be computed in polynomial time for any given history h. □

C Implementation

To implement the VSI framework, we introduce one main practical adaptation: a mixed-integer linear programming (MILP)-based procedure for computing $\widehat{\Phi}(\mathcal{V})$.

In the theoretical algorithm, each value set $\widehat{\Phi}(\mathcal{V})(s)$ is obtained by exhaustively checking every grid point in the bounded value region. Although this approach suffices to establish tractability,

Input: State $s \in S$; current value-set function $\mathcal{V}$ in vertex representation
Output: Vertex representation of $\widehat{\Phi}(\mathcal{V})(s)$

Solve an LP to find an arbitrary $\mathbf{v} \in \Phi(\mathcal{V})(s)$;

if *LP infeasible* **then**
  return $\varnothing$;

Initialize $\mathcal{K}$ as the set of vertices of the grid cell containing $\mathbf{v}$;

repeat
  Compute the convex hull of $\mathcal{K}$;
  for *each facet of the hull* **do**
    Let $\mathbf{n}$ be the outward normal vector of the facet;
    Solve the MILP: $\max_{\mathbf{g} \in G_\xi} \mathbf{n} \cdot \mathbf{g}$, subject to $\mathbf{g} \in \mathrm{neigh}_\xi\big(\Phi(\mathcal{V})(s)\big)$; // $G_\xi = \{\xi \mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^2\}$
    if *MILP returns a grid point* $\mathbf{g}$ *outside the current hull* **then**
      $\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathbf{g}\}$;
      break;
until *no new grid point is found*

return $\mathcal{K}$.

Alg 7. An MILP-based practical algorithm for computing vertices of $\widehat{\Phi}(\mathcal{V})(s)$.

it can be inefficient in practice because many grid points—particularly those in the interior of the polytope—are unnecessary for its vertex representation. Our implementation instead employs an iterative MILP-based separation procedure that searches directly for grid points that expand the current convex hull.

C.1 Vertex Computation via MILP Separation

By definition,

$$\widehat{\Phi}(\mathcal{V})(s) = \mathrm{GridHull}_\xi\big(\mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))\big)\,,$$

so computing $\widehat{\Phi}(\mathcal{V})(s)$ amounts to computing the convex hull of the grid points contained in $\mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$. Under a vertex representation,[10] it suffices to identify a set of grid points whose convex hull equals $\widehat{\Phi}(\mathcal{V})(s)$.

To avoid exhaustively enumerating all candidate grid points, our implementation builds on the LP formulation from Section 6.3. For vertex discovery, we use the same linear constraints that characterize membership in $\mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$, but additionally restrict the candidate value to G_ξ using integer grid indices and optimize a linear objective over the resulting feasible grid points. This yields an MILP that can search directly for grid points outside the current convex hull. The search procedure, presented in Alg. 7, works as follows.

1. **Initialization.** First, find an arbitrary point $\mathbf{v} \in \Phi(\mathcal{V})(s)$ by solving the LP from Section 6.3 without the ξ-neighborhood relaxation. (Effectively, solve Eqs. (10) to (12) for $\tilde{\mathbf{v}}$.) If the LP is infeasible, then $\Phi(\mathcal{V})(s) = \varnothing$ and hence $\widehat{\Phi}(\mathcal{V})(s) = \varnothing$, which we return. Otherwise, we use a grid cell containing $\mathbf{v}$ as the initial polytope.

   For every vertex $\mathbf{g}$ of this cell, $\|\mathbf{g} - \mathbf{v}\|_\infty \le \xi$. Since $\mathbf{v} \in \Phi(\mathcal{V})(s)$, we therefore have

   $$\mathbf{g} \in \mathrm{neigh}_\xi(\Phi(\mathcal{V})(s)) \cap G_\xi.$$

   Hence, by the definition of $\mathrm{GridHull}_\xi$, the entire initial cell is contained in $\widehat{\Phi}(\mathcal{V})(s)$. Using the cell ensures that the initial polytope is full-dimensional, so that the subsequent facet-enumeration procedure is well-defined.

[10] We use "vertex representation" slightly loosely to allow redundant points that are not necessarily extreme points. An irredundant representation can be obtained by extracting the vertices of $\mathrm{conv}(\mathcal{K})$ at the end of Alg. 7, but this is not necessary for the algorithm.

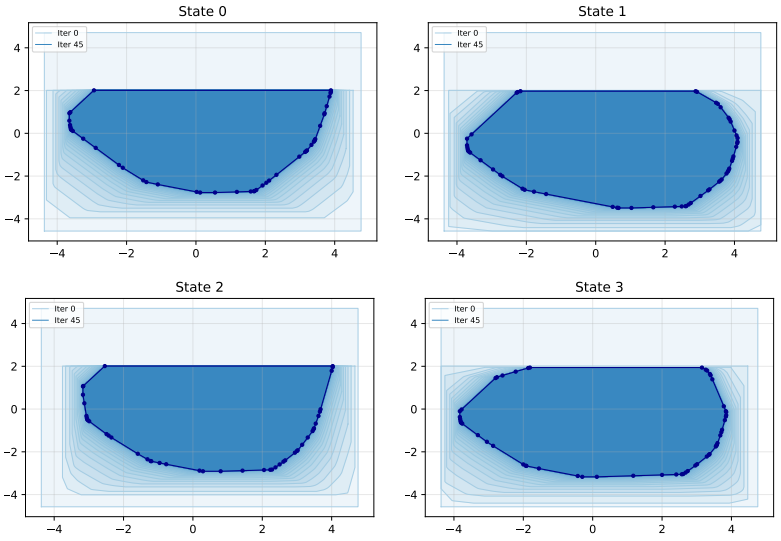


Figure 3: Value sets from an example run. The horizontal and vertical axes show expected discounted reward and cost, respectively. In each plot, the dark inner polytope represents the final value set, with its vertices marked. The lighter outer polytopes represent intermediate value sets generated during the iterations. For this instance, VSI converges after 37 iterations.

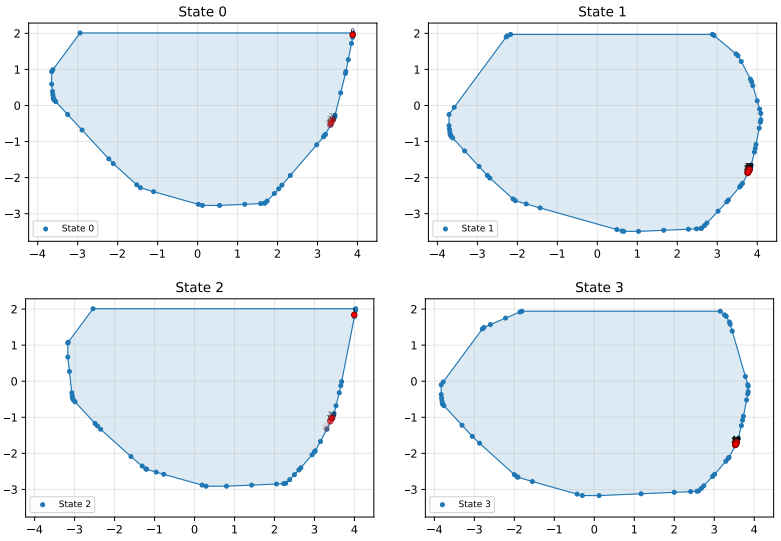


Figure 4: Trajectory of target values generated during the execution of an (ϵ, δ)-optimal policy for the same instance as in Figure 3. The horizontal and vertical axes show expected discounted reward and cost, respectively. The red points are the target values, with lighter shades corresponding to later time steps.

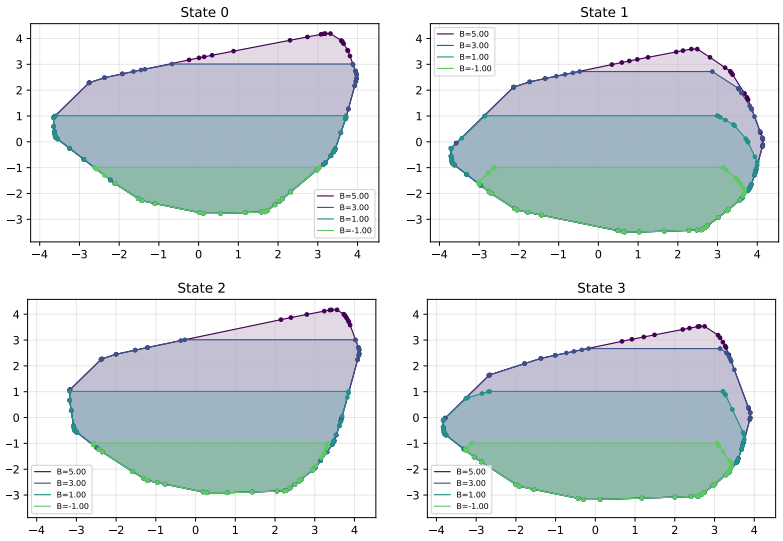


Figure 5: Achievable value sets as the cost bound decreases, for the same instance as in Figure 3. The horizontal and vertical axes show expected discounted reward and cost, respectively. The cost bound in Figures 3 and 4 is $B = 2.0$.

2. **Vertex discovery via facet separation.** The algorithm then enumerates the facets of the current polytope. For each facet, it solves an MILP to optimality in the corresponding outward normal direction over all feasible grid points. The MILP uses the constraints from Eqs. (9) to (12), together with integer variables that restrict the candidate value to G_ξ.
   - If the maximizing grid point lies strictly beyond the current facet, that point is added to the maintained point set, the convex hull and its facets are recomputed, and the procedure repeats.
   - If optimization over every facet direction produces no point outside the current polytope, the current polytope is returned as $\widehat{\Phi}(\mathcal{V})(s)$.

The algorithm terminates because $\mathrm{neigh}_\xi(\Phi(\mathcal{V})(s))$ is bounded and therefore contains only finitely many grid points, while every non-terminal iteration adds a feasible grid point lying outside the current convex hull. The returned polytope is correct because a point outside a full-dimensional polytope must violate at least one of its facet-defining inequalities. At termination, optimization in every outward facet direction certifies that no feasible grid point violates any such inequality. Hence, no feasible grid point lies outside the returned polytope, and the procedure recovers the convex hull of all feasible grid points without exhaustively enumerating them.

C.2 Code

Further details and a Python implementation of the VSI algorithm are available in the GitHub repository at https://github.com/jrgan/VSI. Since $\widehat{\Phi}(\mathcal{V})(s)$ can be computed independently for each state s, the implementation supports parallel computation across states to improve efficiency.

Figures 3 to 5 present results for an instance with 4 states and 5 actions. These results can be reproduced using the code provided in the repository.

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 main claims in the abstract and introduction correspond directly to results stated in lemmas and theorems presented in the paper, with proofs in the appendix. We do not claim results beyond the stated setting, and the introduction makes clear the scope under which the results hold.

   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: The paper discusses limitations of the work in the conclusion section.

   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: The paper states the assumptions for each theoretical result and provides complete proofs in the appendix. The main text includes informal proof sketches to convey intuition, while the appendix contains the full derivations and justifications.

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 main claims are theoretical and supported by proofs in the appendix. The numerical results in Appendix C illustrate the algorithm but do not serve as empirical evidence for those claims.

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: [Yes].

   Justification: The public repository linked in Appendix C provides the Python implementation and code for reproducing the illustrative numerical results. No external datasets are used.

   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 reports no training, test evaluation, or empirical comparison. The numerical results in Appendix C illustrate the algorithm.

   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: Since we do not report experimental comparisons or make claims about average performance, variance, or statistical significance, error bars are not applicable.

   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 reports no empirical performance study; the numerical results in Appendix C are illustrative.

   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: Yes, we conform to 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: This is a theoretical planning study with no direct deployment or application-specific evaluation. Any societal effects would depend on how the framework is applied in future work.

    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 releases algorithmic code, but no high-risk model or dataset for which release safeguards are needed.

    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: No such asset is used in the paper.

    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: [Yes].

    Justification: The new asset is the Python implementation of VSI. Appendix C describes its main algorithmic components and links to the public code repository.

    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 doesn't involve crowdsourcing experiments.

    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: This research 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.