Recursive Constraints to Prevent Instability in Constrained Reinforcement Learning

Jaeyoung Lee*
University of Waterloo
Waterloo, ON, Canada
jaeyoung.lee@uwaterloo.ca

Sean Sedwards*
University of Waterloo
Waterloo, ON, Canada
sean.sedwards@uwaterloo.ca

Krzysztof Czarnecki
University of Waterloo
Waterloo, ON, Canada
krzysztof.czarnecki@uwaterloo.ca

ABSTRACT

We consider the challenge of finding a deterministic policy for a Markov decision process that uniformly (in all states) maximizes one reward subject to a probabilistic constraint over a different reward. Existing solutions do not fully address our precise problem definition, which nevertheless arises naturally in the context of safety-critical robotic systems. This class of problem is known to be hard, but the combined requirements of determinism and uniform optimality can create learning instability.

In this work, after describing and motivating our problem with a simple example, we present a suitable constrained reinforcement learning algorithm that prevents learning instability, using recursive constraints. Our proposed approach admits an approximative form that improves efficiency and is conservative w.r.t. the constraint.

KEYWORDS

Constrained Markov decision process, constrained reinforcement learning, uniform optimality, learning instability

1 INTRODUCTION

Constrained optimization of Markov decision processes is a well-studied field, with a number of algorithms in existence for specific problem definitions [1–7, 10]. In our work, we wish to find a deterministic policy that uniformly maximizes one reward subject to a probabilistic constraint over a different reward. This problem arises naturally in safety-critical systems, such as autonomous driving, where it is required to maximize performance while bounding the probability of hazards, in all states. Since these systems are our focus, we will refer to the notion of satisfying a constraint as safety. Hence, our optimality can be intuitively stated as: in every state, an agent should choose a safe action that maximizes performance or, if no safe action exists, the least unsafe action. Despite its apparent simplicity and similarity to other problems, this definition appears not to have an adequate existing solution.

The characteristics that distinguish our problem specification from previous work is that we require both a deterministic policy (for explainability) and uniform optimality (optimality in every state). Moreover, to ensure safety, the calculation of probability cannot be discounted, as it can with other rewards. In what follows, we show that with naive reinforcement learning algorithms, this combination entails learning instability (oscillation) and inaccurate estimates of probability.

* Contributed equally.

Proc. of the 1st Multi-Objective Decision Making Workshop (MODeM 2021), Hayes, Mannion, Vamplew (eds.), July 14-16, 2021, Online, http://modem2021.cs.nuigalway.ie. 2021.

Much previous work has considered stochastic policies [1, 3, 6, 10], while that which considers deterministic policies [4, 5, 7] does not include the notion of uniform optimality. Unpublished preprints that appear to address a similar problem formulation to ours [8, 11] use examples that do not demonstrate the inherent instability that we have discovered. This is perhaps understandable: even with an adequate example, we have observed that a suitable choice of hyperparameters with a naive learning approach may sufficiently mask the instability for it to seem like mere stochasticity. This is likely to be especially true when using function approximation, where the expectation of convergence is less.

Our proposed solution (Section 5) is a model-based reinforcement learning algorithm that uses recursive constraints to prevent instability while achieving our notion of optimality. Our proposed approach admits an approximative form that improves efficiency and is conservative w.r.t. safety. We suggest this algorithm as a suitable candidate for further extension to the model-free case.

The remainder of this paper proceeds as follows. In Section 2 we define the mathematical preliminaries required for the sequel. In Section 3 we describe the (sometimes conflicting) requirements of our notion of constrained optimality. In Section 4 we give a simple motivating example that demonstrates the inherent instability with naive learning approaches. In Section 5 we describe our algorithm based on recursive constraints and provide results of experiments in Section 6 that demonstrate its advantages. We briefly conclude our work in Section 7.

2 PRELIMINARIES

In this paper, $\overline{\mathbb{N}} := \mathbb{N} \cup \{\infty\}$ and $\overline{\mathbb{N}}_0 := \mathbb{N}_0 \cup \{\infty\}$ denote the sets of extended natural numbers and extended non-negative integers, respectively. For $n, m \in \overline{\mathbb{N}}_0$, we also denote

$$[n..m] := \{k \in \overline{\mathbb{N}}_0 \mid n \leq k \leq m\}$$

We consider a finite Markov decision process (MDP)

$$\mathcal{M} := (\mathcal{S}^+, \mathcal{A}^+, \mathcal{T}, \gamma, \mathcal{R})$$

where $\mathcal{S}^+ := \mathcal{S} \cup \mathcal{S}_\perp$ is a finite set of states consisting of the disjoint sets of all non-terminal states $\mathcal{S}$ and all terminal states $\mathcal{S}_\perp$. $\mathcal{A}^+$ is a finite set of actions and $\gamma \in [0, 1)$ is discount rate; transition function $\mathcal{T}(s, a)$ describes the distribution of next state over $\mathcal{S}^+$, given a current state $s \in \mathcal{S}$ and a chosen action $a \in \mathcal{A}^+$; the reward model $\mathcal{R} : \mathcal{S}^+ \times \mathcal{A}^+ \times \mathcal{S}^+ \to \mathbb{R}$ determines the reward $\mathcal{R}(s, a, s')$ for transition $(s, a, s') \in \mathcal{S} \times \mathcal{A}^+ \times \mathcal{S}^+$ and the terminal one $\mathcal{R}(s, a, s)$ at terminal state-action $sa \in \mathcal{S}_\perp \times \mathcal{A}^+$. The sets $\mathcal{S}^+$ and $\mathcal{A}^+$ are finite. We denote $\mathcal{A}(s)$ ($\subseteq \mathcal{A}^+$) the set of all available actions in state $s \in \mathcal{S}^+$, which is assumed non-empty for all $s \in \mathcal{S}^+$.

A *path* is a sequence of alternating states, actions and rewards

$$(s_0 a_0 r_0)(s_1 a_1 r_1) \cdots (s_{T-1} a_{T-1} r_{T-1}) s_T a_T r_T$$

s.t. $s_t a_t \in \mathcal{S} \times \mathcal{A}(s_t)$, $s_{t+1} \sim \mathcal{T}(s_t, a_t)$ and $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$ for each $t \in [\![0..T-1]\!]$, $s_T a_T \in \mathcal{S}_\perp \times \mathcal{A}(s_T)$ and $r_T = \mathcal{R}(s_T, a_T, s_T)$, where $T \in \overline{\mathbb{N}}_0$ denotes the terminal index, the first hitting time on $\mathcal{S}_\perp$. The reward sequence $r_0 r_1 \cdots r_T$ and discount rate γ define the return

$$R_T := r_0 + \gamma \cdot r_1 + \gamma^2 \cdot r_2 + \gamma^3 \cdot r_3 + \cdots + \gamma^T \cdot r_T \tag{1}$$

A *policy* is a mapping $\pi : \mathcal{S}^+ \to \mathcal{A}^+$ s.t. $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}^+$. Given $s \in \mathcal{S}^+$ (resp. $sa \in \mathcal{S}^+ \times \mathcal{A}(s)$), policy π and MDP $\mathcal{M}$ generate paths s.t. $s_0 = s$ (resp. $s_0 a_0 = sa$) and $a_t = \pi(s_t)$ thereafter, thus inducing probability measures over all such paths. For notational simplicity, we adopt the notations

$$\mathbb{P}(\varphi \mid s_0 = s,\ \pi) \text{ and } \mathbb{P}(\varphi \mid s_0 a_0 = sa,\ \pi),$$

denoting the probabilities that the paths generated by policy π, given $s_0 = s$ and $s_0 a_0 = sa$, respectively, satisfy the property φ;

$$\mathbb{E}(x \mid s_0 = s,\ \pi) \text{ and } \mathbb{E}(x \mid s_0 a_0 = sa,\ \pi)$$

denote the corresponding expectations of a random variable x.

Probabilistic Reachability of Failure States. Let $\mathcal{F}_\perp \subseteq \mathcal{S}_\perp$ be a set of all failure states, then given policy π defines the (unbounded) probabilistic reachability of $\mathcal{F}_\perp$ from $s \in \mathcal{S}^+$:

$$P(s;\pi) := \mathbb{P}(s_T \in \mathcal{F}_\perp \mid s_0 = s,\ \pi)$$

meaning the probability of reaching a failure state $\in \mathcal{F}_\perp$ at the terminal instant T, given that an episode starts from the state $s_0 = s$ and follows the policy π. $P(s;\pi)$ thus quantifies how safe it is to follow the policy π from the initial state s. By definition,

$$P(s;\pi) = \mathbf{1}(s \in \mathcal{F}_\perp) \qquad \forall s \in \mathcal{S}_\perp$$

where $\mathbf{1}(\cdot)$ is the indicator function.

Given a safety threshold $\theta \in [0, 1)$, we define the sets of all safe and unsafe states, $\mathcal{S}(\theta;\pi)$ and $\mathcal{F}(\theta;\pi)$, respectively, as

$$\mathcal{S}(\theta;\pi) := \{s \in \mathcal{S}^+ \mid P(s;\pi) \le \theta\}$$
$$\mathcal{F}(\theta;\pi) := \{s \in \mathcal{S}^+ \mid P(s;\pi) > \theta\}$$

The entire state space $\mathcal{S}^+$ is then partitioned by these disjoint safe and unsafe regions as $\mathcal{S}^+ = \mathcal{S}(\theta;\pi) \cup \mathcal{F}(\theta;\pi)$.

Value Functions. Given policy π, define its value function V as

$$V(s;\pi) := \mathbb{E}(R_T \mid s_0 = s,\ \pi).$$

where R_T is the return (1). The value $V(s;\pi)$ is a performance metric to be optimized at each $s \in \mathcal{S}$. At each terminal state $s \in \mathcal{S}_\perp$, it is directly given by $V(s;\pi) = \mathcal{R}(s, \pi(s), s)$. Similarly, we define the action-value functions for each $(s, a) \in \mathcal{S}^+ \times \mathcal{A}(s)$ as

$$Q(s, a;\pi) := \mathbb{E}(R_T \mid s_0 a_0 = sa,\ \pi)$$
$$\mathcal{P}(s, a;\pi) := \mathbb{P}(s_T \in \mathcal{F}_\perp \mid s_0 a_0 = sa, \pi)$$

which are the same as $Q(s;\pi)$ and $P(s;\pi)$, except that they represent the value and probabilistic reachability, respectively, when the action a is taken at the initial state $s \in \mathcal{S}$ and then π is followed.

3 CONSTRAINED OPTIMALITY

We denote $\hat{\pi}_\theta$ the assumed existent optimal policy that holds the following properties, labelled **P1**—**P4**, associated with threshold $\theta \in [0, 1)$. For notational convenience, we denote

$$\hat{\mathcal{S}}(\theta) := \mathcal{S}(\theta;\hat{\pi}_\theta) \text{ and } \hat{\mathcal{F}}(\theta) := \mathcal{F}(\theta;\hat{\pi}_\theta)$$

P1 *$\hat{\pi}_\theta$ is uniformly optimal in the sense that for any policy π,*

$$P(s;\pi) \le P(s;\hat{\pi}_\theta) \implies V(s;\pi) \le V(s;\hat{\pi}_\theta) \ \ \forall s \in \hat{\mathcal{S}}(\theta) \tag{2}$$
$$V(s;\hat{\pi}_\theta) \le V(s;\pi) \implies P(s;\hat{\pi}_\theta) \le P(s;\pi) \ \ \forall s \in \hat{\mathcal{F}}(\theta) \tag{3}$$

In other words, **P1** means that $\hat{\pi}_\theta$ is Pareto optimal w.r.t. performance and safety, uniformly in its safe and unsafe regions $\hat{\mathcal{S}}(\theta)$ and $\hat{\mathcal{F}}(\theta)$, respectively.

There may exist multiple optimal policies that all satisfy **P1** but achieve different Pareto efficiency. The next property limits such optimality to the case(s) where the safety is maximally improved over the unsafe region.

P2 *$\hat{\pi}_\theta$ is uniformly least unsafe over $\hat{\mathcal{F}}(\theta)$ in the sense that for any policy π s.t. $\pi = \hat{\pi}_\theta$ over $\hat{\mathcal{S}}(\theta)$,*

$$P(s;\hat{\pi}_\theta) \le P(s;\pi) \qquad \forall s \in \hat{\mathcal{F}}(\theta) \tag{4}$$

P2 makes sense also in practice since we trade-off safe and performance within the safe region but *not in the unsafe region*, in which safety comes to be the first priority to be optimized.

Next, it is desirable to have the following monotonicity property among the optimal policies w.r.t. different thresholds.

P3 *If $0 \le \vartheta \le \vartheta' \le 1$, then*

$$V(s;\hat{\pi}_\vartheta) \le V(s;\hat{\pi}_{\vartheta'}) \quad \forall s \in \hat{\mathcal{S}}(\vartheta')$$
$$P(s;\hat{\pi}_\vartheta) \le P(s;\hat{\pi}_{\vartheta'}) \quad \forall s \in \mathcal{S}^+ \tag{5}$$

The intuition behind **P3** is that the weaker the constraint is (i.e., the larger θ), the less conservative the optimal policy is (i.e., the larger probabilistic reachability, thus potentially the better performance). Equation (5) means $\hat{\pi}_{\vartheta'}$ is less conservative than $\hat{\pi}_\vartheta$. In addition, we can see that if **P3** is true, then so are (2) for $\theta = \vartheta'$ and $\pi = \hat{\pi}_\vartheta$, and (3) for $\theta = \vartheta$ and $\pi = \hat{\pi}_{\vartheta'}$, in **P1**. That is, the properties **P1** and **P3** are coherent.

For the next property, let the Bellman operator $\mathcal{T}_\theta$, on the space of functions from each $(s, a) \in \mathcal{S}^+ \times \mathcal{A}(s)$ to $\mathbb{R}^2$, be defined as

$$\mathcal{T}_\theta(Q, \mathcal{P}) := (Q', \mathcal{P}')$$

where $Q'(s, a) = \mathcal{R}(s, a, s)$ and $\mathcal{P}'(s, a) = \mathbf{1}(s \in \mathcal{F}_\perp)$ for all $s \in \mathcal{S}_\perp$, and otherwise,

$$Q'(s, a) = \mathbb{E}\big[r_0 + \gamma \cdot Q(s_1, \pi'(s_1)) \,\big|\, s_0 a_0 = sa\big]$$
$$\mathcal{P}'(s, a) = \mathbb{E}\big[\mathcal{P}(s_1, \pi'(s_1)) \,\big|\, s_0 a_0 = sa\big]$$

for a policy π' given by

$$\pi'(s) \in \begin{cases} \underset{a \in \mathcal{A}_\theta(s;\mathcal{P})}{\arg\max}\ Q(s, a) & \text{if } \mathcal{A}_\theta(s;\mathcal{P}) \neq \varnothing \\ \underset{a \in \mathcal{A}(s)}{\arg\min}\ \mathcal{P}(s, a) & \text{otherwise} \end{cases} \tag{6}$$

$$\mathcal{A}_\theta(s;\mathcal{P}) := \{a \in \mathcal{A}(s) \mid \mathcal{P}(s, a) \le \theta\} \tag{7}$$

where $\mathcal{A}_\theta(s;\mathcal{P})$ is the constrained action set constructed from $\mathcal{P}$, and the dependency of Q', $\mathcal{P}'$, and π' on θ is implicit.

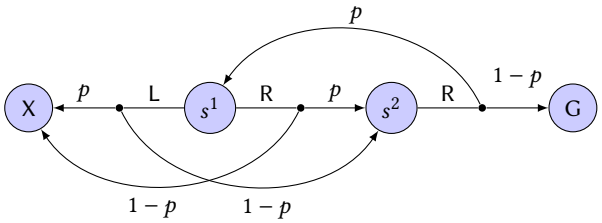


Figure 1: A simple counter-MDP for P4 when $0.5 < p < 1$

In what follows, we state properties regarding the Bellman operator $\mathcal{T}_\theta$ and the optimal action-value functions denoted by

$$\hat{Q}_\theta(s,a) := Q(s,a;\hat{\pi}_\theta) \text{ and } \hat{\mathcal{P}}_\theta(s,a) := \mathcal{P}(s,a;\hat{\pi}_\theta)$$

P4 *$(\hat{Q}_\theta, \hat{\mathcal{P}}_\theta)$ is a fixed point of $\mathcal{T}_\theta$, i.e., $(\hat{Q}_\theta, \hat{\mathcal{P}}_\theta) = \mathcal{T}_\theta(\hat{Q}_\theta, \hat{\mathcal{P}}_\theta)$.*

Note that **P4** is a reasonable property of optimality—if it is true, then the action $\hat{a} = \hat{\pi}_\theta(s)$ at state $s \in \hat{\mathcal{S}}(\theta)$ yields the best value among those actions a satisfying the constraint $\hat{\mathcal{P}}_\theta(s,a) \leq \theta$ and at the unsafe state $s \in \hat{\mathcal{F}}(\theta)$, $\hat{a}$ is safer than any other actions $a \in \mathcal{A}(s)$. The latter is also consistent with **P2**, and **P4** is necessary for convergence of dynamic programming and reinforcement learning methods.

4 LEARNING INSTABILITY

In this section, we present a simple example—the "counter-MDP" shown in Figure 1—which demonstrates that **P4** (the fixed point property) does not hold for certain values of threshold θ. Dynamic programming is therefore unstable.

The state space of this example is $\mathcal{S}^+ = \{s^1, s^2, \mathsf{X}, \mathsf{G}\}$, with $\mathcal{S} = \{s^1, s^2\}$, $\mathcal{S}_\perp = \{\mathsf{X}, \mathsf{G}\}$ and $\mathcal{F}_\perp = \{\mathsf{X}\}$. The action space $\mathcal{A}^+ = \mathcal{A}(s^1) = \{\mathsf{L}, \mathsf{R}\}$, with $\mathcal{A}(s^2) = \{\mathsf{R}\}$. For each transition during an execution there is a reward of -1. Intuitively, the learning objective is to minimize the expected discounted path length, while keeping the probability of reaching X less than or equal to θ. In general, the smaller θ, the higher the probability of the agent reaching G.

We assume the initial state is s^1, where the agent chooses to move either to the left (action L) or to the right (action R). With probability $p > 0$ the agent moves in its chosen direction and with probability $1 - p$ it moves in the opposite direction. For simplicity, the agent has no choice in state s^2 and must choose action R. From s^2, the agent will reach the goal G with probability $1 - p$, or return to state s_1 with probability p. An episode terminates when the agent reaches a terminal state in $\mathcal{S}_\perp$.

Only two policies exist—choose L in s^1 or choose R in s^1—which we denote by π_L and π_R, respectively. For notational simplicity, we denote the corresponding action-value functions in state s^1 by

$$Q_{a\mathsf{L}} := Q(s^1, a; \pi_\mathsf{L}) \qquad Q_{a\mathsf{R}} := Q(s^1, a; \pi_\mathsf{R})$$
$$\mathcal{P}_{a\mathsf{L}} := \mathcal{P}(s^1, a; \pi_\mathsf{L}) \qquad \mathcal{P}_{a\mathsf{R}} := \mathcal{P}(s^1, a; \pi_\mathsf{R})$$

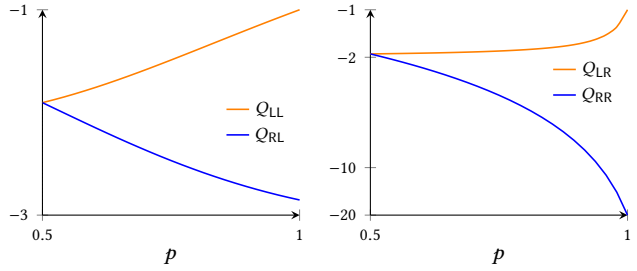
where the dependency on s^1 is implicit. Hence, for example, $Q_{a\mathsf{L}}$ is the Q-function when the agent initially takes $a \in \{\mathsf{L}, \mathsf{R}\}$ at s^1 and then follows π_L. Given the simplicity of the MDP, $Q_{a\mathsf{L}}$ and $\mathcal{P}_{a\mathsf{L}}$ can be given as explicit functions of probability p:

$$\mathcal{P}_{\mathsf{LL}} = \frac{p}{1-pq} \qquad \mathcal{P}_{\mathsf{RL}} = 1 - \frac{pq}{1-pq}$$
$$Q_{\mathsf{LL}} = -\frac{1+\gamma q}{1-\gamma^2 pq} \qquad Q_{\mathsf{RL}} = -\frac{1+\gamma p+\gamma^2 p(p-q)}{1-\gamma^2 pq}$$

where $q := 1 - p$. Similarly,

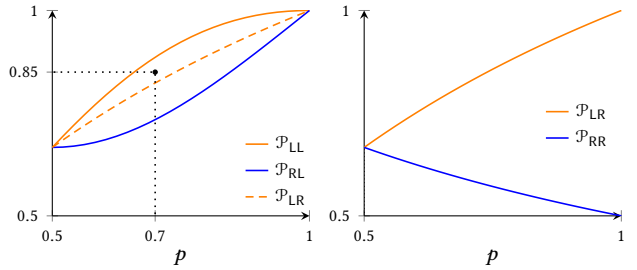
$$\mathcal{P}_{\mathsf{LR}} = 2 \cdot \frac{p}{p+1} \qquad \mathcal{P}_{\mathsf{RR}} = \frac{1}{p+1}$$
$$Q_{\mathsf{LR}} = -\frac{1+\gamma(1-2p)}{1-\gamma p} \qquad Q_{\mathsf{RR}} = -\frac{1}{1-\gamma p}$$

In Figures 2 and 3 we plot these equations against p, for $\gamma = 0.95$. In the following analysis, to demonstrate the counter-example, we consider only $0.5 < p < 1$.



(a) $Q_{a\mathsf{L}}$ **vs** p **(b)** $Q_{a\mathsf{R}}$ **vs** p

Figure 2: Q-functions for s^1 w.r.t. (a) π_L and (b) π_R, for $\gamma = 0.95$



(a) $\mathcal{P}_{a\mathsf{L}}$ **vs** p **(b)** $\mathcal{P}_{a\mathsf{R}}$ **vs** p

Figure 3: P-functions for s^1 w.r.t. (a) π_L and (b) π_R

Figures 2a and 2b plot the Q-values of each action in state s^1, given the policy is π_L and π_R, respectively. We see that regardless of the policy and the value of $p > 0.5$, action L always has the greater Q-value. Hence, while the actual Q-value depends on the policy, there is never any ambiguity about which action to choose to maximize the reward: if the constraint is satisfied, the agent should always choose action L.

Figures 3a and 3b plot the $\mathcal{P}$-values of each action in state s^1, given the policy is π_L and π_R, respectively. As in the case of the Q-values, we see that action L for $p > 0.5$ unambiguously has greater $\mathcal{P}$-value, with the actual $\mathcal{P}$-value depending on the policy. In contrast to the Q-value case, however, the probability that the

$\mathcal{P}$-value represents must also satisfy the constraint, i.e., be less than or equal to threshold θ. The relative performance of the two actions L and R is therefore not sufficient to decide which one is optimal.

To make this decision, we first note that the $\mathcal{P}$-value for action L under π_{L} ($\mathcal{P}_{\mathsf{LL}}$) is the *true* probability for taking action L, and the $\mathcal{P}$-value for action R under π_{R} ($\mathcal{P}_{\mathsf{RR}}$) is the true probability for taking action R. We then call the $\mathcal{P}$-value for action L under π_{R} ($\mathcal{P}_{\mathsf{RL}}$) the *estimated* probability for taking action L (under π_{R}), and call the $\mathcal{P}$-value for action R under π_{L} ($\mathcal{P}_{\mathsf{LR}}$) the estimated probability of action R (under π_{L}).

In Figure 3a we see that the true probability of action L is always greater than the estimated probability of action L. If we choose a threshold θ between the true and estimated values, such that $\mathcal{P}_{\mathsf{LL}} > \theta > \mathcal{P}_{\mathsf{LR}}$, we find that a simple learning agent will not be able to decide which action is optimal using only the $\mathcal{P}$-values. Suppose, during the learning process, the current policy is π_{L}, which we know maximizes the Q-value, the agent will see that the $\mathcal{P}$-value ($\mathcal{P}_{\mathsf{LL}}$) is greater than θ, which does not satisfy the constraint. Figure 3a shows that the estimated value of action R under π_{L} ($\mathcal{P}_{\mathsf{LR}}$) will satisfy the constraint, so the agent chooses π_{R}. Under π_{R}, however, the estimated value of action L ($\mathcal{P}_{\mathsf{LR}}$) appears now to satisfy the constraint, so the (memoryless) agent chooses π_{L} once again.

To see this phenomenon in a concrete learning example, consider the policy iteration process described in Table 1, with $(p, \theta, \gamma) = (0.7, 0.85, 0.95)$ and π_{R} as its initial policy. At the first iteration $i = 1$, $\mathcal{P}_{\mathsf{LR}} \leq \theta$ indicates that action L seems to be safe, and thus we choose π_{L}. However, at the next iteration ($i = 2$), $\mathcal{P}_{\mathsf{LL}} \not\leq \theta$ shows that π_{L} is *not* safe, which forces us to choose π_{R} again. Unfortunately, $\mathcal{P}_{\mathsf{LR}} \leq \theta$ at iteration $i = 2$ falsely indicates that action L is safe, again! This oscillation continues ad infinitum.

In summary, we see that a naive learning approach with the example shown in Figure 1 will not reach a fixed point, so **P4** does not hold in general.

5 RECURSIVE CONSTRAINTS

In this section, we answer to the following questions—(i) "what is wrong with fixed point property **P4**?"; (ii) "how can we solve it?"

To address the former, we point out the mismatch between **P1** and **P4**. If $\hat{\pi}_\theta$ holds (2) in **P1**, then it satisfies

$$\hat{\pi}_\theta(s) \in \arg\max_{a \in \hat{\mathcal{A}}_\theta(s)} \hat{Q}_\theta(s,a) \qquad \forall s \in \hat{\mathcal{S}}(\theta) \tag{8}$$
$$\hat{\mathcal{A}}_\theta(s) := \{a \in \mathcal{A}(s) \mid \hat{\mathcal{P}}_\theta(s,a) \leq P(s\,; \hat{\pi}_\theta)\}$$

Intuitively, (8) means that $\hat{\pi}_\theta$ must yield a higher value over the safe region, $\hat{\mathcal{S}}(\theta)$, than any of its conservative one-point modifications in $\hat{\mathcal{S}}(\theta)$. However, we can easily notice the difference between the two constrained action sets—$\hat{\mathcal{A}}_\theta(s)$ in (8) and $\mathcal{A}_\theta(s\,; \hat{\mathcal{P}}_\theta)$, where $\mathcal{A}_\theta(s\,; \cdot)$ is defined by (7) and used to construct the policy (7) and thus the Bellman operator $\mathcal{T}_\theta$ in **P4**. Here, $\hat{\mathcal{P}}_\theta(s,a)$ is constrained by the threshold θ in the latter but by $P(s\,; \hat{\pi}_\theta)$ in the former. For each $s \in \hat{\mathcal{S}}(\theta)$, since $P(s\,; \hat{\pi}_\theta) \leq \theta$ holds by the definition of $\hat{\mathcal{S}}(\theta)$, the former $\hat{\mathcal{A}}_\theta(s)$ is more conservative than the latter $\mathcal{A}_\theta(s\,; \hat{\mathcal{P}}_\theta)$, that is, $\hat{\mathcal{A}}_\theta(s) \subseteq \mathcal{A}_\theta(s\,; \hat{\mathcal{P}}_\theta)$. Therefore, **P1** $\not\Rightarrow$ **P4**, in general.

In a similar manner to (8), $\hat{\pi}_\theta$ is safer over $\hat{\mathcal{F}}(\theta)$ than any of its less performing one-point modifications in $\hat{\mathcal{F}}(\theta)$ if it satisfies (3) in **P1**. In this case, we have

$$\hat{\pi}_\theta(s) \in \arg\min_{a \in \hat{\mathcal{A}}'_\theta(s)} \hat{\mathcal{P}}_\theta(s,a) \qquad \forall s \in \hat{\mathcal{F}}(\theta) \tag{9}$$
$$\hat{\mathcal{A}}'_\theta(s) := \{a \in \mathcal{A}(s) \mid \hat{Q}_\theta(s,a) \leq V(s\,; \hat{\pi}_\theta)\}$$

On the other hand, if $\hat{\pi}_\theta$ holds **P2**, then it satisfies (9) with $\hat{\mathcal{A}}'_\theta(s)$ replaced by $\mathcal{A}(s)$ shown in the policy (6) on the unsafe argmin-part. Therefore, from the standard dynamic programming theory, we can conclude that **P1** and **P2** imply **P4** if the constrained action set $\mathcal{A}_\theta(s\,; \hat{\mathcal{P}}_\theta)$ is equal to or replaced by $\hat{\mathcal{A}}_\theta(s)$.

From the discussions above on the conservatism of **P1** w.r.t. **P4**, we hypothesize that

> *$\mathcal{A}_\theta(\cdot\,; \hat{\mathcal{P}}_\theta)$ has to be replaced with a constrained action set, e.g., $\hat{\mathcal{A}}_\theta(\cdot)$, that is more conservative, for* **P4** *to be true.*

In fact, the hypothesis is true for the counter-MDP in Figure 1. To see this, we revisit the policy iteration example with $(p, \theta, \gamma) = (0.7, 0.85, 0.95)$ and π_{R} as its initial policy, but also with recursive constraints illustrated in Table 2. Here, the meaning of the recursive constraints is clear from Table 2—for each action $a \in \{\mathsf{L}, \mathsf{R}\}$, we superimpose all the constraints on a by recursion, up to the current iteration, and use it to judge whether a is a safe action or not. Denoting $\mathrm{C}_a(i)$ such a recursive constraint for action a, made at iteration i, then at each iteration in Table 2, it satisfies by its construction

$$\mathrm{C}_{\mathsf{L}}(1) = (\mathcal{P}_{\mathsf{LR}} \leq \theta)$$
$$\mathrm{C}_{\mathsf{L}}(2) = (\mathcal{P}_{\mathsf{LL}} \not\leq \theta) \wedge \mathrm{C}_{\mathsf{L}}(1) = (\mathcal{P}_{\mathsf{LL}} \not\leq \theta) \wedge (\mathcal{P}_{\mathsf{LR}} \leq \theta)$$
$$\mathrm{C}_{\mathsf{L}}(3) = (\mathcal{P}_{\mathsf{LR}} \leq \theta) \wedge \mathrm{C}_{\mathsf{L}}(2) = (\mathcal{P}_{\mathsf{LR}} \leq \theta) \wedge (\mathcal{P}_{\mathsf{LL}} \not\leq \theta)$$
$$\mathrm{C}_{\mathsf{L}}(4) = (\mathcal{P}_{\mathsf{LR}} \leq \theta) \wedge \mathrm{C}_{\mathsf{L}}(3) = (\mathcal{P}_{\mathsf{LR}} \leq \theta) \wedge (\mathcal{P}_{\mathsf{LL}} \not\leq \theta)$$
$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

From this, we observe that the constraint C_{L} on the action a at s^1 is now stabilized and thereby, yields the same safe policy π_{R} from iteration $i = 3$, as shown in Table 2, whereas it was not without such recursive constraints, as illustrated in Table 1.

On the other hand, the idea of recursive constraints, demonstrated in Table 2, still has the issue that except for policy iteration, a dynamic programming or reinforcement learning method typically does not wait until its value function (e.g., $\mathcal{P}(s, a\,; \pi)$) is accurately estimated. In Table 2 for example, if $\mathcal{P}_{\mathsf{LL}}$ and/or $\mathcal{P}_{\mathsf{LR}}$ is not correctly estimated at the previous iterations, then the recursive constraint C_{L} at the current iteration can be so deteriorated and messy as it is constructed from all the previous constraints, including those based on inaccurate predictions at the early stages. In particular, the initial values of $\mathcal{P}_{\mathsf{LL}}$ and $\mathcal{P}_{\mathsf{LR}}$ are typically random and has no information, which introduces and transfers absurd random constraints, to all iterations. Hence, the recursive constraints at and around the initial stages must be also stabilized, in order to generalize themselves to a broad class of reinforcement learning methods.

In order to solve such a remaining issue on stability, we (i) replace the iteration axis in Table 2 with the axis of horizon window $n = 1, 2, \ldots, N$ and (ii) replace the constraint at stage n with

$$\max_{m \in [1..n]} \hat{\mathcal{P}}^m(s,a) \leq \theta$$

Table 1: Policy iteration on the MDP in Figure 1 for $(p, \theta, \gamma) = (0.7, 0.85, 0.95)$

Iteration i		1	2	3	4	5	$\cdots$
Given policy		π_{R}	π_{L}	π_{R}	π_{L}	π_{R}	$\cdots$
Constraints	L	$\mathcal{P}_{\mathsf{LR}} \approx 0.82 \leq \theta = 0.85$	$\mathcal{P}_{\mathsf{LL}} \approx 0.89 \not\leq \theta$	$\mathcal{P}_{\mathsf{LR}} \leq \theta$	$\mathcal{P}_{\mathsf{LL}} \not\leq \theta$	$\mathcal{P}_{\mathsf{LR}} \leq \theta$	$\cdots$
	R	$\mathcal{P}_{\mathsf{RR}} \approx 0.59 \leq \theta = 0.85$	$\mathcal{P}_{\mathsf{RL}} \approx 0.73 \leq \theta$	$\mathcal{P}_{\mathsf{RR}} \leq \theta$	$\mathcal{P}_{\mathsf{RL}} \leq \theta$	$\mathcal{P}_{\mathsf{RR}} \leq \theta$	$\cdots$

Table 2: Policy iteration with recursive constraints on the MDP in Figure 1 for $(p, \theta, \gamma) = (0.7, 0.85, 0.95)$

Iteration i		1	2	3	4	$\cdots$
Given policy		π_{R}	π_{L}	π_{R}	π_{R}	$\cdots$
Constraints	L	$\mathrm{C}_{\mathsf{L}} \leftarrow (\mathcal{P}_{\mathsf{LR}} \leq \theta)$	$\mathrm{C}_{\mathsf{L}} \leftarrow (\mathcal{P}_{\mathsf{LL}} \not\leq \theta) \wedge \mathrm{C}_{\mathsf{L}}$	$\mathrm{C}_{\mathsf{L}} \leftarrow (\mathcal{P}_{\mathsf{LR}} \leq \theta) \wedge \mathrm{C}_{\mathsf{L}}$	$\mathrm{C}_{\mathsf{L}} \leftarrow (\mathcal{P}_{\mathsf{LR}} \leq \theta) \wedge \mathrm{C}_{\mathsf{L}}$	$\cdots$
	R	$\mathrm{C}_{\mathsf{R}} \leftarrow (\mathcal{P}_{\mathsf{RR}} \leq \theta)$	$\mathrm{C}_{\mathsf{R}} \leftarrow (\mathcal{P}_{\mathsf{RL}} \leq \theta) \wedge \mathrm{C}_{\mathsf{R}}$	$\mathrm{C}_{\mathsf{R}} \leftarrow (\mathcal{P}_{\mathsf{RR}} \leq \theta) \wedge \mathrm{C}_{\mathsf{R}}$	$\mathrm{C}_{\mathsf{R}} \leftarrow (\mathcal{P}_{\mathsf{RR}} \leq \theta) \wedge \mathrm{C}_{\mathsf{R}}$	$\cdots$

where $\hat{\mathcal{P}}^n(s,a)$ is for an (over-)approximation of the n-bounded probabilistic reachability

$$\mathcal{P}^n(s,a\,;\pi) \coloneqq \mathbb{P}(s_{\min(T,n)} \in \mathcal{F}_\perp \,|\, s_0 a_0 = sa, \pi)$$

w.r.t. the policy $\pi = \hat{\pi}^{n-1}$ obtained at the previous stage $n-1$. Here, an over-approximation means $0 \leq \mathcal{P}^n(s,a\,;\hat{\pi}^{n-1}) \leq \hat{\mathcal{P}}^n(s,a) \leq 1$. To describe our proposal, we also denote

$$P^n(s\,;\pi) \coloneqq \mathbb{P}(s_{\min(T,n)} \in \mathcal{F}_\perp \,|\, s_0 = s, \pi)$$

which satisfies $P^n(s\,;\pi) = \mathcal{P}^n(s,\pi(s)\,;\pi)$ for all $s \in \mathcal{S}^+$.

Note that $\mathcal{P}^n(\cdot\,;\pi)$ at horizon $n=1$ is now stable since it does not depend on the policy anymore, as shown below:

$$\hat{\mathcal{P}}^1(s,a) \coloneqq \mathcal{P}^1(s,a\,;\pi) = \mathbb{P}(s_1 \in \mathcal{F}_\perp | \, s_0 a_0 = sa) \quad \forall s \in \mathcal{S}$$

From $\hat{\mathcal{P}}^1(\cdot)$, we construct the first policy $\hat{\pi}^1$ on the horizon axis as (in this case, substitute $n=1$)

$$\hat{\pi}^n(s) \in \begin{cases} \arg\max_{a \in \hat{\mathcal{A}}^n(s)} \hat{Q}^n(s,a) & \text{if } \hat{\mathcal{A}}^n(s) \neq \varnothing \\ \arg\min_{a \in \mathcal{A}(s)} \hat{\mathcal{P}}^n(s,a) & \text{otherwise} \end{cases} \tag{10}$$

$$\hat{\mathcal{A}}^n(s) \coloneqq \{a \in \mathcal{A}(s) \,|\, \mathrm{C}_a(n\,;s)\} \qquad \hat{Q}^n(s,a) \coloneqq Q(s,a\,;\pi^n)$$

where the constraint $\mathrm{C}_a(1\,;s) \coloneqq (\hat{\mathcal{P}}^1(s,a) \leq \theta)$, and the dependency on the threshold θ is all implicit. Also note that the horizons of $\hat{Q}^1$ and $\hat{\mathcal{P}}^1$ are ∞ and 1, respectively.

At the next stage $n=2$, note that the n-bounded probabilistic reachability w.r.t. $\hat{\pi}^1$ satisfies the Bellman equation of the form

$$\mathcal{P}^2(s,a\,;\hat{\pi}^1) = \mathbb{E}\big[\hat{\mathcal{P}}^1(s_1, \hat{\pi}^1(s_1)) \,\big|\, s_0 a_0 = sa\big] \qquad \forall s \in \mathcal{S} \tag{11}$$

Therefore, denoting $\hat{\mathcal{P}}^2(s,a) \coloneqq \mathcal{P}^2(s,a\,;\hat{\pi}^1)$, the second policy $\hat{\pi}^2$ can be easily constructed via (10), whose constraint is recursively defined as $\mathrm{C}_a(2\,;s) \coloneqq (\hat{\mathcal{P}}^2(s,a) \leq \theta) \wedge \mathrm{C}_a(1\,;s)$.

At horizon $n = 3, 4, 5, \ldots, N$, the n-bounded probabilistic reachability $\mathcal{P}^n(\cdot\,;\hat{\pi}^{n-1})$ w.r.t. the policy $\hat{\pi}^{n-1}$ given at the previous step $n-1$ satisfies the Bellman equation:

$$\mathcal{P}^n(s,a\,;\hat{\pi}^{n-1}) = \mathbb{E}\big[P^{n-1}(s_1\,;\hat{\pi}^{n-1}) \,\big|\, s_0 a_0 = sa\big] \qquad \forall s \in \mathcal{S}$$

However, in order to obtain $P^{n-1}(\cdot\,;\hat{\pi}^{n-1})$, we need to calculate $P^m(\cdot\,;\hat{\pi}^{n-1})$ and use it in the backward induction for $P^{m+1}(\cdot\,;\hat{\pi}^{n-1})$, all the way through $m = 1, 2, 3, \ldots, n-1$. The longer the horizon n is, the more complexity this procedure induces in space and time.

Instead, our design choice is to use a substitute $\hat{\mathcal{P}}^{n-1}(s, \hat{\pi}^{n-1}(s))$ obtained at the previous stage $n-1$. Therefore, we define

$$\hat{\mathcal{P}}^n(s,a) \coloneqq \mathbb{E}\big[\hat{\mathcal{P}}^{n-1}(s_1\,;\hat{\pi}^{n-1}(s_1)) \,\big|\, s_0 a_0 = sa\big] \quad \forall s \in \mathcal{S}$$

and construct the policy $\hat{\pi}^n$ at the current horizon n via (10), w.r.t. the recursive constraint

$$\mathrm{C}_a(n\,;s) \coloneqq (\hat{\mathcal{P}}^n(s,a) \leq \theta) \wedge \mathrm{C}_a(n-1\,;s)$$

Here, $\hat{\mathcal{P}}^{n-1}(s, \hat{\pi}^{n-1}(s))$ typically over-approximates $P^{n-1}(s\,;\hat{\pi}^{n-1})$ since $\hat{\pi}^{n-1}$ is constructed with a fewer constraints than $\hat{\pi}^{n-2}$, and

$$\begin{cases} P^{n-1}(s\,;\hat{\pi}^{n-1}) & = \mathcal{P}^{n-1}(s, \hat{\pi}^{n-1}(s)\,;\hat{\pi}^{n-1}) \\ \hat{\mathcal{P}}^{n-1}(s, \hat{\pi}^{n-1}(s)) & = \mathcal{P}^{n-1}(s, \hat{\pi}^{n-1}(s)\,;\hat{\pi}^{n-2}) \end{cases}$$

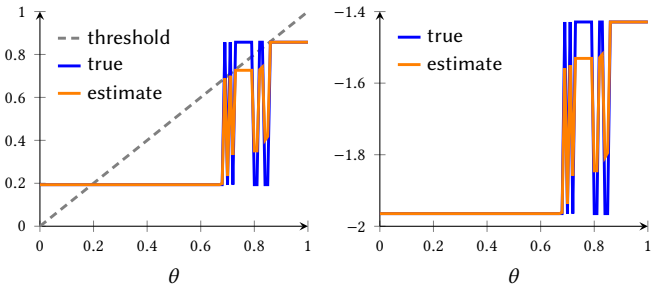
Finally, we provide the policy π^N at the last horizon N as the receding-horizon solution that is potentially conservative but able to uniformly and (sub-)optimally improves the performance subject to N-bounded probabilistic reachability constraint imposed on every state. To address the instability issue, the final policy π^N has the recursive constraints $\mathrm{C}_a(N\,;s)$ that contain all constraints w.r.t. shorter horizons, i.e.,

$$\mathrm{C}_a(N\,;s) = \bigwedge_{n \in [1..N]} (\hat{\mathcal{P}}^n(s,a) \leq \theta) \iff \Big(\max_{n \in [1..N]} \hat{\mathcal{P}}^n(s,a)\Big) \leq \theta$$

where each $\hat{\mathcal{P}}^n(\cdot)$ is recursively defined from the initial one $\hat{\mathcal{P}}^1(\cdot)$ that is independent of any policy and hence can be stably obtained.

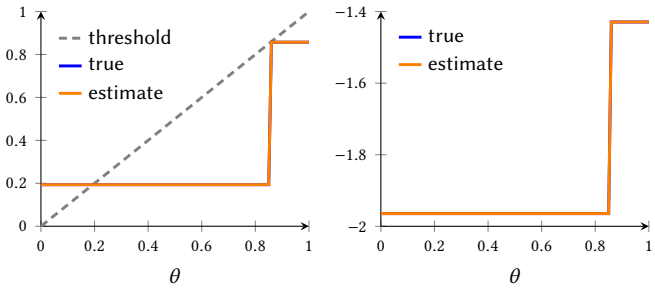
Since the recursive constraint $\mathrm{C}_a(N\,;s)$ makes the constrained action set $\hat{\mathcal{A}}^N(s)$ monotonically decreasing as N increases, and since $\hat{\mathcal{A}}^N(s)$ is finite, it is stabilized within a finite horizon, say M, after which the process becomes the same as that for unconstrained MDPs but w.r.t. the action space $\hat{\mathcal{A}}^M(\cdot)$. That is, the final policy π^N converges as $N \to \infty$, at the infinite-horizon.

Now that we have a well-defined solution π^N, the next question is "how to find it?". To be precise, the remaining issue is "how to estimate all the action-value functions $\hat{\mathcal{P}}^n$ (n-horizon) and $\hat{Q}^n$ (infinite-horizon) for $n = 1, 2, 3, \ldots, N$?". The answer is that reinforcement learning ideas, such as value iteration and Q-learning, can be employed for such a purpose. In what follows, we compare naive value iteration based on the Bellman operator $\mathcal{T}_\theta$, which does not hold the fixed point property **P4**, with the idea of recursive



(a) $P(s_0 ; \hat{\pi})$ **vs** θ **(b)** $V(s_0 ; \hat{\pi})$ **vs** θ

Figure 4: Experimental results for naive value iteration



(a) $P(s_0 ; \hat{\pi})$ **vs** θ **(b)** $V(s_0 ; \hat{\pi})$ **vs** θ

Figure 5: Experimental results of our proposed approach

constraints and bounded probabilistic reachability presented in this section.

6 EXPERIMENTS

To illustrate the benefits of our approach, we implemented competing value iteration methods and applied them to a "cliffworld" environment [9] that generalizes the MDP in Figure 1 by allowing actions in all possible directions; transitions to the desired direction are made with probability 0.5 and to a random direction with the remaining 0.5 probability. Using the full range of θ, in Figure 4 we present the results of naive value iteration (Algorithm 1) with a total of 50 iterations. In Figure 5 we present corresponding results using our proposed approach (Algorithm 2) with a total of 15 iterations and horizon $N = 15$. Figure 4 shows that within a wide range around $\theta = 0.8$: (i) naive value iteration thinks that the current policy is safe (orange curve) while it actually is not (blue curve); (ii) the switching between the policies randomly induces errors, with the values not converging. In contrast, Figure 5 shows that our approach does not exhibit any such chattering or violations. More experimental and theoretical study is ongoing work, but we can clearly see from the present experiments that the solution of our approach for each θ is apparently stable and converged.

7 CONCLUSION

In this work, we have considered a constrained optimization problem that arises naturally in the context of safety-critical systems. Despite this, and our problem's apparently reasonable requirements, we find that there is no existing approach to adequately address it. On closer inspection, we have discovered that our insistence on a deterministic policy and uniform optimality imposes restrictions that make an already hard task [5] more difficult. In particular, we have shown with a simple example that a naive learning approach can be unstable and have no fixed point. We have thus proposed and demonstrated a model-based reinforcement learning algorithm that uses recursive constraints to address the instability. The approximative form of this approach reduces computational cost, while making the constraints conservative.

In future work, we will adapt our algorithm to the model-free case, and explore the possibility of using our approach with function approximation.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support of Japanese Science and Technology agency (JST) ERATO project JPMJER1603: HASUO Metamathematics for Systems Design. The authors also express their gratitude to Abhinav Grover, for his participation in early discussions and implementation.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). MLR Press, 22–31.
[2] Eitan Altman. 1999. *Constrained Markov Decision Processes.* CRC Press.
[3] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. 2018. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 31. OpenReview, 8103–8112.
[4] Dmitri Dolgov and Edmund Durfee. 2005. Stationary Deterministic Policies for Constrained MDPs with Multiple Rewards, Costs, and Discount Factors. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 1326–1331.
[5] Eugene A Feinberg. 2000. Constrained Discounted Markov Decision Processes and Hamiltonian Cycles. *Mathematics of Operations Research* 25, 1 (2000), 130–140.
[6] Eugene A Feinberg and Adam Shwartz. 1999. Constrained Dynamic Programming with Two Discount Factors: Applications and an Algorithm. *IEEE Trans. Automat. Control* 44, 3 (1999), 628–631.
[7] Peter Geibel and Fritz Wysotzki. 2005. Risk-Sensitive Reinforcement Learning Applied to Control Under Constraints. *Journal of Artificial Intelligence Research* 24 (2005), 81–108.
[8] Gabriel Kalweit, Maria Huegle, Moritz Werling, and Joschka Boedecker. 2020. Deep Constrained Q-learning. arXiv:2003.09398
[9] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction.* MIT press.
[10] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. 2019. Reward Constrained Policy Optimization. In *International Conference on Learning Representations (ICLR)*. OpenReview.
[11] Aditya Undurti, Alborz Geramifard, and Jonathan P. How. 2011. Function Approximation for Continuous Constrained MDPs. (2011).

Algorithm 1: Naive Value Iteration

Input:
$\mathcal{M}$: MDP $(\mathcal{S}^+, \mathcal{A}^+, \mathcal{T}, \gamma, \mathcal{R})$ k : number of iterations
$\mathcal{F}_\perp$: set of all failure states
θ : constraint threshold $\in [0, 1)$

Output:
π : solution policy
$Q, \mathcal{P}$: estimates of action-value functions $Q(\cdot, \pi)$ and $\mathcal{P}(\cdot, \pi)$

/* initialization */

$Q(s,a) \leftarrow$ initial value, e.g. 0 $\quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}(s)$

$\mathcal{P}(s,a) \leftarrow$ initial value $\in [0,1] \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}(s)$

$Q(s,a) \leftarrow \mathcal{R}(s,a,s) \quad \forall (s,a) \in \mathcal{S}_\perp \times \mathcal{A}(s)$

$\mathcal{P}(s,a) \leftarrow \mathbf{1}(s \in \mathcal{F}_\perp) \quad \forall (s,a) \in \mathcal{S}_\perp \times \mathcal{A}(s)$

/* main loop */

repeat k **times**

  $\hat{\mathcal{A}}(s) \leftarrow \{a \in \mathcal{A}(s) \mid \mathcal{P}(s,a) \leq \theta\} \quad \forall s \in \mathcal{S}^+$

  $\pi \leftarrow \text{GetPolicy}(\hat{\mathcal{A}}, Q, \mathcal{P})$

  foreach $(s,a) \in \mathcal{S} \times \mathcal{A}(s)$ **do**

    $\hat{Q}(s,a) \leftarrow \sum_{s' \in \mathcal{S}^+} \mathcal{T}(s,a)(s') \cdot \big(\mathcal{R}(s,a,s') + \gamma \cdot Q(s', \pi(s'))\big)$

    $\hat{\mathcal{P}}(s,a) \leftarrow \sum_{s' \in \mathcal{S}^+} \mathcal{T}(s,a)(s') \cdot \mathcal{P}(s', \pi(s'))$

  $Q \leftarrow \hat{Q}$

  $\mathcal{P} \leftarrow \hat{\mathcal{P}}$

/* update the solution policy */

$\hat{\mathcal{A}}(s) \leftarrow \{a \in \mathcal{A}(s) \mid \mathcal{P}^n(s,a) \leq \theta\} \quad \forall s \in \mathcal{S}^+$

$\pi \leftarrow \text{GetPolicy}(\hat{\mathcal{A}}, Q, \mathcal{P})$

return $\pi, Q, \mathcal{P}$

Subroutine 1: GetPolicy$(\hat{\mathcal{A}}, Q, \mathcal{P})$

foreach $s \in \mathcal{S}^+$ **do**

$$\pi(s) \leftarrow a \in \begin{cases} \arg\max_{a' \in \hat{\mathcal{A}}(s)} Q(s,a') & \text{if } \hat{\mathcal{A}}(s) \neq \varnothing \\ \arg\min_{a' \in \mathcal{A}(s)} \mathcal{P}(s,a') & \text{otherwise} \end{cases}$$

return π

Algorithm 2: Value Iteration with Recursive Constraints

Input:
$\mathcal{M}$: MDP $(\mathcal{S}^+, \mathcal{A}^+, \mathcal{T}, \gamma, \mathcal{R})$ k : number of iterations
$\mathcal{F}_\perp$: set of all failure states N : horizon $\in \mathbb{N}$
θ : constraint threshold $\in [0, 1)$

Output:
π : solution policy
$Q^N, \mathcal{P}^N$: estimates of functions $Q(\cdot, \pi)$ and $\mathcal{P}^N(\cdot, \pi)$

/* initialization */

$Q^{1:N}(s,a) \leftarrow$ initial value, e.g. 0 $\quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}(s)$

$\mathcal{P}^1(s,a) \leftarrow \sum_{s' \in \mathcal{F}_\perp} \mathcal{T}(s,a)(s') \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}(s)$

$Q^{1:N}(s,a) \leftarrow \mathcal{R}(s,a,s) \quad \forall (s,a) \in \mathcal{S}_\perp \times \mathcal{A}(s)$

$\mathcal{P}^{1:N}(s,a) \leftarrow \mathbf{1}(s \in \mathcal{F}_\perp) \quad \forall (s,a) \in \mathcal{S}_\perp \times \mathcal{A}(s)$

/* main loop */

repeat k **times**

  $\hat{\mathcal{A}} \leftarrow \mathcal{A}$

  for $n = 1, 2, \ldots, N$ **do**

    $\hat{\mathcal{A}}(s) \leftarrow \{a \in \hat{\mathcal{A}}(s) \mid \mathcal{P}^n(s,a) \leq \theta\} \quad \forall s \in \mathcal{S}^+$

    $\pi \leftarrow \text{GetPolicy}(\hat{\mathcal{A}}, Q^n, \mathcal{P}^n)$

    foreach $(s,a) \in \mathcal{S} \times \mathcal{A}(s)$ **do**

      $\hat{Q}(s,a) \leftarrow \sum_{s' \in \mathcal{S}^+} \mathcal{T}(s,a)(s') \cdot \big(\mathcal{R}(s,a,s') + \gamma \cdot Q^n(s', \pi(s'))\big)$

    foreach $(s,a) \in \mathcal{S} \times \mathcal{A}(s)$ **if** $n < N$ **do**

      $\mathcal{P}^{n+1}(s,a) \leftarrow \sum_{s' \in \mathcal{S}^+} \mathcal{T}(s,a)(s') \cdot \mathcal{P}^n(s', \pi(s'))$

    $Q^n \leftarrow \hat{Q}$

/* update the solution policy */

$\hat{\mathcal{A}} \leftarrow \mathcal{A}$

for $n = 1, 2, \ldots, N$ **do**

  $\hat{\mathcal{A}}(s) \leftarrow \{a \in \hat{\mathcal{A}}(s) \mid \mathcal{P}^n(s,a) \leq \theta\} \quad \forall s \in \mathcal{S}^+$

$\pi \leftarrow \text{GetPolicy}(\hat{\mathcal{A}}, Q^N, \mathcal{P}^N)$

return $\pi, Q^N, \mathcal{P}^N$