Sauté RL: Almost Surely Safe Reinforcement Learning Using State Augmentation

Aivar Sootla [1] **Alexander I. Cowen-Rivers** [1 2] **Taher Jafferjee** [1] **Ziyan Wang** [1] **David Mguni** [1] **Jun Wang** [3] **Haitham Bou-Ammar** [1 4]

Abstract

Satisfying safety constraints almost surely (or with probability one) can be critical for the deployment of Reinforcement Learning (RL) in real-life applications. For example, plane landing and take-off should ideally occur with probability one. We address the problem by introducing Safety Augmented (Sauté) Markov Decision Processes (MDPs), where the safety constraints are eliminated by augmenting them into the state-space and reshaping the objective. We show that Sauté MDP satisfies the Bellman equation and moves us closer to solving Safe RL with constraints satisfied almost surely. We argue that Sauté MDP allows viewing the Safe RL problem from a different perspective enabling new features. For instance, our approach has a plug-and-play nature, i.e., any RL algorithm can be "Sautéed". Additionally, state augmentation allows for policy generalization across safety constraints. We finally show that Sauté RL algorithms can outperform their state-of-the-art counterparts when constraint satisfaction is of high importance.

1. Introduction

Reinforcement Learning (RL) offers a great framework for solving sequential decision-making problems using interactions with an environment (Sutton & Barto, 2018). In this context, safety constraint satisfaction and robustness are vital properties for the successful deployment of RL algorithms. Safe RL has received significant attention in recent years, but many unsolved challenges remain, e.g., constraint satisfaction during and after training, efficient algorithms, etc. While different types of constraints were considered in the past, e.g. averaged over trajectory, CVaR, and chance constraints (Chow et al., 2017), satisfying constraints almost surely (or with probability one) received less attention to date. This problem is quite important as many safety-critical applications require constraint satisfaction almost surely. For example, we ideally would like to guarantee a safe plane landing and take-off with probability one, while landing with a probability of 0.99 may not be sufficient. Similarly, an autonomous vehicle should be able to stay in the lane with probability one, while keeping this constraint "on average" can potentially lead to catastrophic consequences.

We represent the safety constraints as a (discounted) sum of nonnegative costs bounded from above by (what we call) *a safety budget*. As the safety costs are accumulated during an episode, there is less freedom in choosing a safe trajectory and hence the available safety budget diminishes over time. We can treat the remaining safety budget as a new state quantifying the risk of constraint violation. This idea can be traced back to classical control methods of augmenting the safety constraints into the state-space cf. (Daryin & Kurzhanski, 2005), however, we adapt it to stochastic problems and RL applications. First, we reshape the objective to take into account the remaining safety budget by assigning infinite values if the budget was spent. Thus we obtain Safety AUgmenTEd (Sauté) Markov Decision Process (MDP). Incorporating the constraint into the objective enforces this constraint for every controlled trajectory which leads to constraint satisfaction with probability one. Furthermore, Sauté MDP satisfies the Bellman equation justifying the use of *any critic-based RL method*. Finally, since the value functions now additionally depend on the safety information, the cost-to-go in our setting implicitly includes information about possible safety constraint violations.

Employing the state augmentation for safe RL has been explored in the past. For example, (Calvo-Fullana et al., 2021) augmented the Lagrangian multiplier into the state space, while keeping the Lagrangian objective. The Lagrangian multiplier contains information about the safety cost that

[1]Huawei R&D UK. [2]Technische Universität Darmstadt. [3]University College London [4]Honorary Lecturer at University College London. Correspondence to: Aivar Sootla <aivar.sootla@huawei.com>, Haitham Bou-Ammar <haitham.ammar@huawei.com>, Jun Wang <jun.wang@cs.ucl.ac.uk>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

could be exploited by the policy. We argue that such a construction is not required as we can track the accumulated safety cost instead. (Chow et al., 2017) augmented the CVaR constraint in the state-space, however, their augmentation had a realization suitable only for the CVaR constraint. They also had to resort to the Lagrangian approach for solving the problem due to (again) the specificity of the CVaR constraint. We discuss in detail these state augmentation methods in Appendix B, but note that (Calvo-Fullana et al., 2021), (Chow et al., 2017) have not extended their methods to modern model-free and model-based RL methods such as trust region policy optimization (TRPO) (Schulman et al., 2015), proximal policy optimization (PPO) (Schulman et al., 2017), soft actor-critic (SAC) (Haarnoja et al., 2018), model-based policy optimization (MBPO) (Janner et al., 2019), probabilistic ensembles with trajectory sampling (PETS) (Chua et al., 2018).

We argue that our state augmentation approach allows viewing the safe RL problem from a different angle. Our state augmentation should be seen as a modification of an environment rather than an RL algorithm. It is implemented as a wrapper around OpenAI gym (Brockman et al., 2016) environments, which allows "sautéing" *any* RL algorithm. While we mostly test with model-free approaches (PPO, TRPO, SAC), the model-based methods are also "sautéable", which we illustrate on MBPO and PETS. We then demonstrate that a policy trained on one safety budget can generalize to other budgets. Further, if we randomly sample the initial safety state (i.e., the safety budget), then we can learn a policy *for all safety budgets* in a set.

Related work. Safe RL is based on the constrained MDP (c-MDP) formalism (Altman, 1999), which has spawned many directions of research. A topic of considerable interest is safe exploration (Turchetta et al., 2016; Koller et al., 2018; Dalal et al., 2018; Wachi et al., 2018; Zimmer et al., 2018; Bharadhwaj et al., 2020), where the goal is to ensure constraint satisfaction while exploring for policy improvement. Another line of research is to use classical control methods and concepts to learn a safe policy (Chow et al., 2018; 2019; Berkenkamp et al., 2017; Ohnishi et al., 2019; Cheng et al., 2019; Akametalu et al., 2014; Dean et al., 2019; Fisac et al., 2019). In these works, the authors also make strong prior assumptions such as partial knowledge of the model, and an initial safe policy to define a problem that can be solved.

Besides classical control other tools were used in safe RL, e.g., a two-player framework with a task agent and a safety agent cooperating to solve the task (Mguni et al., 2021), a curriculum learning approach, where the teacher resets the student violating safety constraints (Turchetta et al., 2020), learning to reset if the safety constraint is violated (Eysenbach et al., 2018).

While these are interesting topics of research, the classical RL setting with minimal assumptions is arguably more common in RL literature. A considerable effort was made in solving safe RL in the model-based setting (Polymenakos et al., 2020; Cowen-Rivers et al., 2022; Kamthe & Deisenroth, 2018). In these works, the model was learned using Gaussian processes (Rasmussen & Williams, 2005; Deisenroth & Rasmussen, 2011) allowing for an effective uncertainty estimation albeit with scalability limitations. Constrained versions of model-free RL algorithms such as TRPO, PPO, and SAC were also developed. For example, CPO (Achiam et al., 2017) generalized TRPO to a constrained case by explicitly adding constraints to the trust region update. (Ray et al., 2019) were the first, to our best knowledge, to implement the Lagrangian version of PPO, SAC, and TRPO. These methods largely followed (Chow et al., 2017), who, however, considered an RL problem with a conditional-value-at risk (CVaR) constraints instead of average constraints. While Chow et al used classical policy gradient algorithms, recently this work was extended to PPO (Cowen-Rivers et al., 2022) and SAC (Yang et al., 2021) algorithms. The Lagrangian method and CPO were improved in (Stooke et al., 2020; Yang et al., 2019), respectively. Finally, (Ding et al., 2020) proposed a natural policy gradient for c-MDPs.

2. Vanilla RL and Safe RL with Constraints

We first review basic RL and MDP concepts and adapt some definitions to our setting.

Definition 1 *We define a Markov Decision Process (MDP) as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, c, \gamma_c \rangle$, where $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; $\gamma_c \in (0, 1)$ is the task discount factor, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, i.e., $\boldsymbol{s}_{t+1} \sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and $c : \mathcal{S} \times \mathcal{A} \to [0, +\infty)$ is the task cost. We associate the following optimization problem with the MDP:*

$$\begin{aligned} &\min_{\pi} \ \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{task}}, \\ &J_{\text{task}} \triangleq \sum_{t=0}^{\infty} \gamma_c^t c(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{a}_t \sim \pi, \end{aligned} \tag{1}$$

where $\mathbb{E}^{\pi}_{\boldsymbol{s}}$ is the mean over the transitions and the policy π.

The objective J_{task} implicitly depends on the initial state $\boldsymbol{s}_0$ distribution and the policy π, but we drop this dependence to simplify notation. Throughout we assume that spaces $\mathcal{S}$ and $\mathcal{A}$ are continuous, e.g., $\mathcal{S} \subset \mathbb{R}^{n_s}$ and $\mathcal{A} \subset \mathbb{R}^{n_a}$, where n_s and n_a are the dimensions of the state and action spaces. We consider an infinite-horizon problem for theoretical convenience, in practice, however, we use the finite horizon (or episodes) setting as common in the RL literature (Sutton & Barto, 2018). We also minimize the objective adapting the notation by (Chow et al., 2017). To solve the problem, the

value functions are typically introduced:

$$V(\pi, \boldsymbol{s}_0) = \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{task}},$$
$$V^*(\boldsymbol{s}) = \min_{\pi} V(\pi, \boldsymbol{s}),$$

where with a slight abuse of notation $\mathbb{E}^{\pi}_{\boldsymbol{s}}$ is not averaging over the initial state $\boldsymbol{s}_0$. In general, finding the representation of the optimal policy is not easy and the optimal policy can depend on the past trajectory, i.e., past state-action pairs. Remarkably, under some mild assumptions (see, for example, Appendix A) the optimal policy solving Equation 1 depends only on the current state, i.e., $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t)$. This is a consequence of the Bellman equation which holds for the optimal value function V^*_{task}:

$$V^*_{\text{task}}(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\mathcal{A}} \left(c(\boldsymbol{s}, \boldsymbol{a}) + \gamma_c \mathbb{E}_{\boldsymbol{s}'\sim p(\cdot|\boldsymbol{s},\boldsymbol{a})} V^*_{\text{task}}(\boldsymbol{s}') \right).$$

Additionally, using the Bellman equation we can also reduce the cost-to-go estimation to a static optimization rather than dynamic[1]. Both of these properties are at the heart of all RL algorithms. Now we can move on to constrained MDPs.

Definition 2 *The constrained MDP (c-MDP) is a tuple $\mathcal{M}_c = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, c, \gamma_c, l, \gamma_l \rangle$ where additional terms are the safety cost $l : \mathcal{S} \times \mathcal{A} \to [0, +\infty)$ and the safety discount factor $\gamma_l \in (0, 1)$. The associated optimization problem is:*

$$\min_{\pi} \ \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{task}} \tag{2a}$$
$$\textit{s. t.:} \ \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}} \geq 0, \tag{2b}$$
$$J_{\text{safety}} \triangleq d - \sum_{t=0}^{\infty} \gamma_l^t l(\boldsymbol{s}_t, \boldsymbol{a}_t). \tag{2c}$$

We will refer to the nonnegative value d as the safety budget.

The average over the accumulated cost can be replaced by another statistic. For example, (Chow et al., 2017) proposed to use conditional value at risk $\text{CVaR}_{\alpha}(X) = \min_{\nu\in\mathbb{R}} \left(\eta + \frac{1}{1-\alpha} \mathbb{E}\, \text{ReLU}(X - \nu) \right)$ for $\alpha \in (0, 1)$:

$$\min_{\pi,\nu} \ \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{task}},$$
$$\text{s. t.:} \ \nu + \frac{1}{1-\alpha} \mathbb{E}^{\pi}_{\boldsymbol{s}} \ \text{ReLU}\left(d - J_{\text{safety}} - \nu\right) \leq d.$$

In both cases, the problem can be solved using the Lagrangian approach, cf. (Chow et al., 2017). For example, in the case of Equation 2 we can rewrite this problem as

$$\min_{\pi} \ \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{task}} - \lambda^* \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}}, \ \text{where}$$
$$\lambda^* = \begin{cases} 0 & \text{if } \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}} \geq 0, \\ +\infty & \text{if } \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}} < 0. \end{cases}$$

[1]There are still some non-stationary components in learning the value functions since the new data is constantly acquired.

Instead of optimizing over an indicator λ^*, one formulates an equivalent min-max problem (Bertsekas, 1997):

$$\min_{\pi} \max_{\lambda \geq 0} \widehat{J} \triangleq \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{task}} - \lambda \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}}.$$

Indeed, for every fixed policy π, the optimal λ is equal to λ^*: if $\mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}}$ is nonnegative then there is no other choice but setting λ to zero, if the $\mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}}$ is negative then maximum over λ is $+\infty$ cf. (Bertsekas, 1997). Thus we obtained the Lagrangian formulation of c-MDP. Now for actor-critic algorithms we need to estimate an additional value function $V_{\text{safety}}(\pi, \boldsymbol{s}_0) = \mathbb{E}^{\pi}_{\boldsymbol{s}} J_{\text{safety}}$ tracking the safety cost. Note that it is not clear if the optimal policy can be found using the commonly used representation $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t)$ since it is not clear what is the equivalent of the Bellman equation in the constrained case. Hence, the common policy representation (i.e., $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t)$) can create some limitations. Even intuitively the actions should depend on the safety constraint in some way, but they do not.

3. Sauté RL: Safety AUgmenTEd Reinforcement Learning

3.1. Main Idea in the Deterministic Case

We start by presenting the main idea in the deterministic case in order to simplify presentation. We consider a c-MDP with deterministic transitions, costs and one constraint:

$$\min_{\pi} \ J_{\text{task}}, \tag{3}$$
$$\text{s. t.:} \ J_{\text{safety}} \geq 0. \tag{4}$$

and its Lagrangian formulation

$$\min_{\pi} \max_{\lambda \geq 0} \widehat{J} \triangleq J_{\text{task}} - \lambda J_{\text{safety}}. \tag{5}$$

We adapt the ideas by (Daryin & Kurzhanski, 2005) to the RL case, and propose to reduce Problem 3 to an MDP. In particular, we remove the constraint by using state augmentation and then by reshaping the cost.

Let us take a step back and note that enforcing the constraint in Equation 3 is equivalent to enforcing the infinite number of the following constraints:

$$\sum_{k=0}^{t} \gamma_l^k l(\boldsymbol{s}_k, \boldsymbol{a}_k) \leq d, \ \forall t \geq 0. \tag{3b*}$$

This is because we assumed that the instantaneous cost is nonnegative and the accumulated safety cost cannot decrease. Therefore, if the constraint is violated at some time t_v, it will be violated for all $t \geq t_v$. It seems counter-intuitive to transform a problem with a single constraint into a problem with an infinite number of constraints. However,

our goal here is to incorporate the constraints into the instantaneous task cost, thus taking into account safety while solving the task. This will be easier to perform while considering the constraint for all times t. To do so we track the remaining safety budget to assess constraint satisfaction at every time step. The remaining safety budget at time t can be computed as $\boldsymbol{w}_t = d - \sum_{m=0}^{t} \gamma_l^m l(\boldsymbol{s}_m, \boldsymbol{a}_m)$, however, we will track a scaled version of $\boldsymbol{w}_t$, namely, $\boldsymbol{z}_t = \boldsymbol{w}_{t-1}/\gamma_l^t$, which has a time-independent update:

$$\begin{aligned}\boldsymbol{z}_{t+1} &= (\boldsymbol{w}_{t-1} - \gamma_l^t l(\boldsymbol{s}_t, \boldsymbol{a}_t))/\gamma_l^{t+1} \\ &= (\boldsymbol{z}_t - l(\boldsymbol{s}_t, \boldsymbol{a}_t))/\gamma_l, \\ \boldsymbol{z}_0 &= \boldsymbol{d}.\end{aligned}$$

Since the variable $\boldsymbol{z}_t$ is Markovian (i.e., $\boldsymbol{z}_{t+1}$ depends only on $\boldsymbol{z}_t$, $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$), we can augment our state-space with the variable $\boldsymbol{z}_t$. Now since we enforce the constraint $\boldsymbol{z}_t \geq 0$ for all times $t \geq 0$, we can reshape the instantaneous task cost to account for the safety constraint:

$$\widetilde{c}(\boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{a}_t) = \begin{cases} c(\boldsymbol{s}_t, \boldsymbol{a}_t) & \boldsymbol{z}_t \geq 0, \\ +\infty & \boldsymbol{z}_t < 0. \end{cases}$$

Now we can formulate the problem *without* constraints

$$\min_{\pi} \sum_{t=0}^{\infty} \gamma_c^t \widetilde{c}(\boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{a}_t). \tag{6}$$

Note that the safety cost l, and the safety discount factor γ_l are now part of the transition. The variable $\boldsymbol{z}_t$ can be intuitively understood as a risk indicator for constraint violation. The policy can learn to tread carefully for some values of $\boldsymbol{z}_t$ thus learning to interpret $\boldsymbol{z}_t$ as the distance to constraint violation. Note that the augmented state by (Chow et al., 2017) tracks a value related to the CVaR computation rather than the remaining safety budget (see Appendix B.2), while the augmented state by (Calvo-Fullana et al., 2021) is the Lagrange multiplier (see Appendix B.3). In both cases, the augmented state by itself is not a very intuitive risk indicator for safety during an episode. Furthermore, in our case the safety budget d is the initial safety state, which enables generalization across safety budgets as we show in our experiments. This was not done by (Calvo-Fullana et al., 2021) and (Chow et al., 2017).

To summarize, we have effectively showed the following:

Theorem 1 *An optimal policy for any of Equations 3, 5 and 6 is also an optimal policy for all of these problems.*

Next, we discuss how to deal with the infinity in the cost function and we generalize this idea to the stochastic case.

3.2. Safety Augmented Markov Decision Processes

The derivation in the general case are fairly similar at first and we obtain the following transition functions as above:

$$\begin{aligned}\boldsymbol{s}_{t+1} &\sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t), & \boldsymbol{s}_0 &\sim \mathcal{S}_0, \\ \boldsymbol{z}_{t+1} &= (\boldsymbol{z}_t - l(\boldsymbol{s}_t, \boldsymbol{a}_t))/\gamma_l, & \boldsymbol{z}_0 &= d.\end{aligned} \tag{7}$$

Note that the transitions in Equation 7 are still Markovian and non-stationary, which simplifies the further algorithm development.

In order to avoid dealing with infinite values in the cost $\widetilde{c}(\boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{a}_t)$, we introduce a proxy problem with a computationally friendlier cost:

$$\widetilde{c}_n(\boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{a}_t) = \begin{cases} c(\boldsymbol{s}_t, \boldsymbol{a}_t) & \boldsymbol{z}_t \geq 0, \\ n & \boldsymbol{z}_t < 0, \end{cases} \tag{8}$$

where n is a hyper-parameter and introduce the Safety AUgmenTEd (Sauté) MDP $\widetilde{\mathcal{M}}_n$.

Definition 3 *Given a c-MDP $\mathcal{M}_c = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, c, \gamma_c, l, \gamma_l \rangle$, we define a Safety Augmented Markov Decision Process (Sauté MDP) as a tuple $\widetilde{\mathcal{M}}_n = \langle \widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{\mathcal{P}}, \widetilde{c}_n, \gamma_c \rangle$, where $\widetilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Z}$; $\widetilde{\mathcal{P}} : \widetilde{\mathcal{S}} \times \mathcal{A} \times \widetilde{\mathcal{S}} \to [0, 1]$ and defined in Equation 7, and $\widetilde{c}_n : \widetilde{\mathcal{S}} \times \mathcal{A} \to [0, +\infty)$. We associate the following problem with Sauté MDP:*

$$\min_{\pi} \mathbb{E} \sum_{t=0}^{\infty} \gamma_c^t \widetilde{c}_n(\boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{a}_t). \tag{9}$$

Now we need to answer two questions: a) what is the optimal representation of π_n^*; b) what is the relation of the MDPs $\widetilde{\mathcal{M}}_n$ and $\widetilde{\mathcal{M}}_\infty$. While the first question appears to be quite straightforward, the second requires revising results from stochastic optimal control. For readers' convenience, we summarize these results here. We make the following assumptions:

A1. The functions $\widetilde{c}_n(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{a})$ are bounded, measurable, nonnegative, and lower semi-continuous on $\widetilde{\mathcal{S}} \times \mathcal{A}$;
A2. $\mathcal{A}$ is compact;
A3. The transition $\mathcal{P}$ is weakly continuous on $\widetilde{\mathcal{S}} \times \mathcal{A}$, i.e., for any continuous and bounded function u on $\widetilde{\mathcal{S}}$ the map $(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{a}) \to \int_{\widetilde{\mathcal{S}}} u(\boldsymbol{x}, \boldsymbol{y}) \mathcal{P}(d\boldsymbol{x}, d\boldsymbol{y} | \boldsymbol{s}, \boldsymbol{z}, \boldsymbol{a})$ is continuous.

Note that these assumptions are rather mild and satisfied in many RL tasks. For instance, the task costs are often continuous and bounded, which is enough to satisfy Assumption 1 for the reshaped costs $\widetilde{c}_n$. Similarly, compactness of the action space is typically assumed in the RL setting as well. Finally, Assumption A3 is satisfied, if, for example, the transition function $\mathcal{P}$ is a Gaussian with continuous mean and variance (cf. (Arapostathis et al., 1993)).

Under these assumptions we can prove the following results (the proof is in Appendix A).

Theorem 2 *Consider a Sauté MDP $\widetilde{\mathcal{M}}_n$ satisfying Assumptions A1-A3 with the associated Equation 9, then:*

a) for any finite n the Bellman equation is satisfied, i.e., there exists a function $V_n^*(\boldsymbol{s}, \boldsymbol{z})$ such that

$$V_n^*(\boldsymbol{s}, \boldsymbol{z}) = \min_{\boldsymbol{a} \in \mathcal{A}} \left(\widetilde{c}_n(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{a}) + \gamma_c \mathbb{E}_{\boldsymbol{s}', \boldsymbol{z}'} V_n^*(\boldsymbol{s}', \boldsymbol{z}') \right),$$

where $\boldsymbol{s}', \boldsymbol{z}' \sim \widetilde{p}(\cdot|\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{a})$. Furthermore, the optimal policy solving $\widetilde{\mathcal{M}}_n$ has the representation $\boldsymbol{a} \sim \pi_n^*(\cdot|\boldsymbol{s}, \boldsymbol{z})$;

b) the optimal value functions V_n^* for $\widetilde{\mathcal{M}}_n$ converge monotonically to V_∞^* — the optimal value function for $\widetilde{\mathcal{M}}_\infty$.

The practical implication of our theoretical result is threefold: a) we can use critic-based methods and guarantee their convergence under standard assumptions, b) the optimal policy is Markovian and depends on the safety budget, i.e., $\boldsymbol{a} \sim \pi_n^*(\cdot|\boldsymbol{s}, \boldsymbol{z})$, and c) vanilla RL methods can be applied to solve $\widetilde{\mathcal{M}}_n$. We finally stress that we can solve $\widetilde{\mathcal{M}}_\infty$ only approximately by solving $\widetilde{\mathcal{M}}_n$ for a large enough n.

3.3. Almost Surely Safe Markov Decision Processes

While we derived an algorithm and studied the theoretical properties of our problem, we are yet to discuss what problem we are aiming to solve. Consider the following formulation for safe reinforcement learning.

Definition 4 *An* almost surely constrained MDP *is a c-MDP $\mathcal{M}_c$ with the associated optimization problem:*

$$\min_{\pi(\cdot|\boldsymbol{s}_t, \boldsymbol{z}_t)} \mathbb{E} J_{\text{task}}, \tag{10a}$$

$$\text{s.t.: } \boldsymbol{z}_t \geq 0 \;\; a.s., \; \forall t \geq 0, \tag{10b}$$

$$\boldsymbol{z}_{t+1} = (\boldsymbol{z}_t - l(\boldsymbol{s}_t, \boldsymbol{a}_t))/\gamma_l, \tag{10c}$$

$$\boldsymbol{z}_0 = d,$$

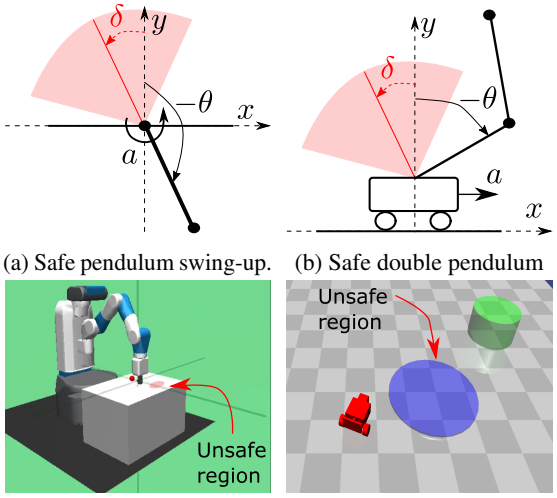
where a.s. stands for "almost surely" (with probability 1).

Solving this problem using RL should deliver almost surely safe policies benefiting safety-critical applications. This formulation is much stronger than the average and the CVaR constrained ones. Tackling Problem 10 directly seems to be impossible at the first sight. Remarkably, solving Sauté MDP $\widetilde{\mathcal{M}}_\infty$ is equivalent to solving Equation 10. The equivalence should be understood in the following sense:

Theorem 3 *Consider a Sauté MDP $\widetilde{\mathcal{M}}_\infty$ and Equation 9. Suppose there exists an optimal policy $\pi^*(\cdot|\boldsymbol{s}_t, \boldsymbol{z}_t)$ solving Equation 9 with a finite cost, then $\pi^*(\cdot|\boldsymbol{s}_t, \boldsymbol{z}_t)$ is an optimal policy for Equation 10.*

Proof: The finite cost in $\widetilde{\mathcal{M}}_\infty$ implies the satisfaction of Constraint 10b almost surely. Now since the policy π^* was obtained by minimizing the same objective as in Equation 10 and satisfies Constraint 10b almost surely, it also minimizes the objective in Equation 10. □

Above we showed that a policy solving Sauté MDP $\widetilde{\mathcal{M}}_\infty$ and Equation 9 is actually policy solving Safe RL almost surely and Equation 10. We further showed that the optimal value function V_n^* for $\widetilde{\mathcal{M}}_n$ converges to the optimal value function V_∞^* for $\widetilde{\mathcal{M}}_\infty$ under some mild conditions. While in many practical scenarios, this means that the policy for $\widetilde{\mathcal{M}}_n$ for large n approximates well the policy solving Safe RL almost surely, there may be some specific settings when this is not the case. Further assumptions can improve our analysis, however, we refer the reader to (Hernández-Lerma & Muñoz de Ozak, 1992) for a detailed discussion.



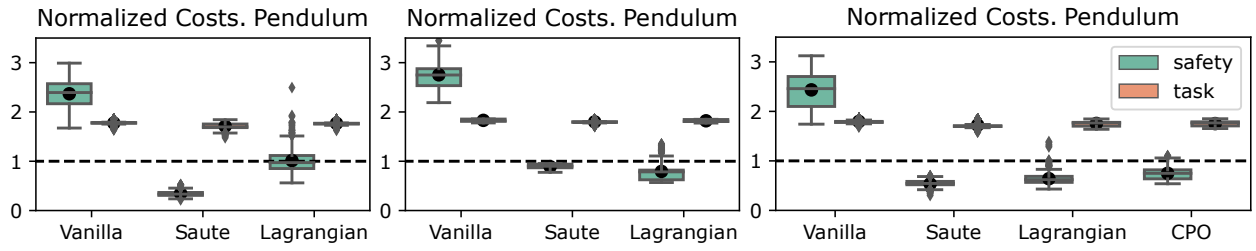
(a) Safe pendulum swing-up. (b) Safe double pendulum

(c) Safe reacher (d) Safety gym

Figure 1. Panels a and b: safe pendulum environments. In both cases, θ - is the angle from the upright position, a is the action, δ - is the unsafe pendulum angle, the safety cost is the distance toward the unsafe pendulum angle, which is incurred only in the red area. Panel e: safe reacher: robot needs to avoid the unsafe region (in red). Panel d: a safety gym environment: robot needs to reach the goal (in green) while avoiding the unsafe region (in blue).

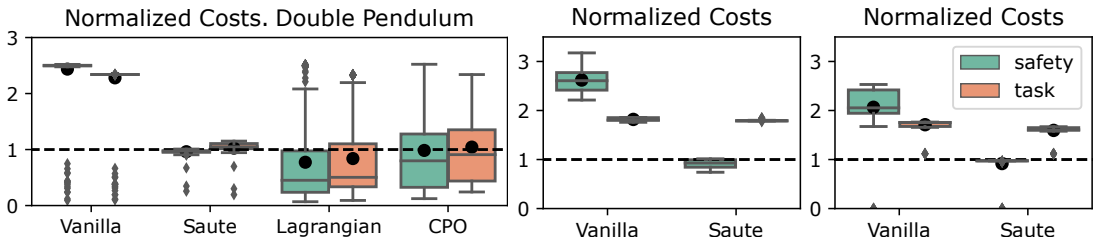
4. Experiments

Environments. We demonstrate the advantages and the limitations of our approach on three OpenAI gym environments with safety constraints (pendulum swing-up, double pendulum balancing, reacher) and the OpenAI safety gym environment (schematically depicted in Figure 1). In the environment design we follow previous work by (Kamthe & Deisenroth, 2018), (Cowen-Rivers et al., 2022), (Yang et al., 2021) and delegate the details to Appendix D.



(a) PPO (b) SAC (c) TRPO

Figure 2. Sauté RL as a plug-n-play approach. Box plots for normalized safety(on the left) and task (on the right) costs for SAC, PPO and TRPO-type algorithms on pendulum swing-up environment with the safety budget 30 after 300 epochs of training. In all figures the task cost are divided by -100 and the safety costs are divided by 30, the dashed lined indicate the safety threshold. In all cases, "sautéed" algorithms deliver safe policies with probability 1 (outliers whiskers do not cross the dashed line), while Lagrangian methods and CPO have trajectories violating the safety constraints. For task costs the higher values are better.



(a) TRPO (b) MBPO (c) PETS

Figure 3. Sauté TRPO on the double pendulum environment (panel a) and Sauté MBRL on the pendulum swing-up environment (Panels b and c). Box plots for safety costs (on the left) and normalized task (on the right). Panel a: The task costs are divided by -80, while the safety costs by 40 with the safety budget 40. Panels b and c: The task costs are divided by 100, while the safety costs by 30 with 30 being the safety budget. In all plots dashed lines indicate the safety threshold. For task costs the higher values are better.

Implementation. The main benefit of our approach to safe RL is the ability to "sauté" *any RL algorithm*. This is because we do not need to change the algorithm itself (besides some cosmetic changes), but create a wrapper around the environment. The implementation is quite straightforward and the only "trick" we used is normalizing the safety state by dividing it by the safety budget:

$$\boldsymbol{z}_{t+1} = (\boldsymbol{z}_t - l(\boldsymbol{s}_t, \boldsymbol{a}_t)/d)/\gamma_l,$$
$$\boldsymbol{z}_0 = 1.$$

Hence the variable $\boldsymbol{z}_t$ is always between zero and one. Note this does not affect our theoretical results. The reset and step functions have to be overloaded to augment the safety state and shape the cost as in Equation 8. More details on "sautéed" environment implementation are available in Appendix C. We used safety starter agents (Ray et al., 2019) as the core implementation for model-free methods, their Lagrangian versions (PPO, TRPO, SAC), and CPO. We use the hyper-parameters listed in Appendix E. For our model-based implementations, we used (Pineda et al., 2021) as the core library, which has PyTorch implementation of MBPO (Janner et al., 2019), and PETS (Chua et al., 2018). Finally, we implemented a CVaR constrained safe RL based on the safety starter agents. We discuss our implementation in Appendix C.2. Our implementations are available online (Sootla et al., 2022).

Evaluation protocols. In all our experiments we used 5 different seeds, we save the intermediate policies and evaluate them on 100 different trajectories in all our figures and tables. One exception is the evaluation of PETS, for which we used 25 trajectories. Note that in all the plots we use returns based on the original task costs c, not the reshaped task costs $\widetilde{c}_n$ to evaluate the performance. In all our experiments we set the safety discount factor for Sauté RL equal to one, while the safety discount factor for other algorithms varies. We also use box-and-whisker plots with boxes showing median, q_3 and q_1 quartiles of the distributions (75th and 25th percentiles, respectively), whiskers depicting the error bounds computed as $1.5(q_3 - q_1)$, as well as outliers, e.g., points lying outside the whisker intervals (Waskom, 2021). We add black dots to the plots which signify the mean. We use box-and-whisker plots so that we can showcase the outliers and the percentiles, which are important criteria for the evaluation of almost surely constraints.

Sauté RL is an effective plug-n-play approach. We first demonstrate that Sauté RL can be easily applied to both on-policy (PPO, TRPO) and off-policy algorithms (SAC)

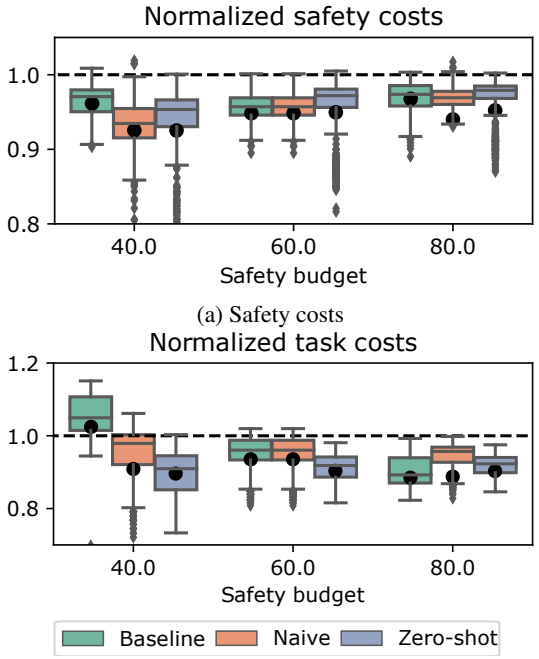
without significant issues. We run all these algorithms on the pendulum swing-up environment. We test the policies with the initial state sampled around the downright position of the pendulum. The results in Figure 2 indicate that PPO, TRPO, and SAC can be effectively "sautéed" and deliver policies safe almost surely (i.e., with probability one and all trajectories satisfy the constraint). Note that the difference in behavior for Trust Region-based algorithms is the smallest, while Sauté SAC delivers the best overall performance. We also present the evaluation during training in Figures A2, A3, A4 and A5 in Appendix.

Sauté Model-Based RL. We proceed by "sautéing" MBRL methods: MBPO and PETS. As the results in Figures 3(b), and 3(c) suggest, we lose some performance, but guarantee safety in both cases. Remarkably we could "sauté" both MPC and policy-based methods without significant issues.

As we have demonstrated the plug-n-play nature of our approach for model-free and model-based methods, in all further experiments we evaluate our method on Trust Region-based algorithms only, i.e., Vanilla TRPO, its variants, and CPO. We did so because Sauté TRPO has a lower gap in performance with Lagrangian TRPO than Sauté SAC and Sauté PPO have with their Lagrangian versions. However, a fair evaluation against CPO was also appealing.

"Safety on average" can be very unsafe even in deterministic environments. While safety on average is less restrictive than safety almost surely, in some situations safety on average can lead to unwanted behaviors. We design the safe double pendulum environment in such a way that task and safety costs are correlated. Hence restricting the safety cost leads to restricting the task cost and forces the Lagrangian algorithm to balance a trade-off between these objectives. Further, the constraints on "average" allow Lagrangian TRPO and CPO to learn the policies that prioritize minimizing task costs for some initial states and minimizing safety costs for other initial states. While the constraint is satisfied on average, the distributions of task and safety costs for both CPO and Lagrangian TRPO have a large variance (see Figure 3(a)). Further, some outliers have similar behavior to the Vanilla TRPO. Sauté TRPO on the other hand achieves the best overall performance. We plot the evaluation curves during training in Appendix in Figure A6.

Generalization across safety budgets. Since the safety budget d enters the problem formulation as the initial value of the safety state, we can generalize to a different safety budget *after training* by changing the initial safety state. We train three separate set of policies for safety budgets 40, 60, 80, we then take the policies trained for the safety budget 60 and evaluate them on the safety budgets 40 and 80. The test results of this *naïve* generalization approach over 5 different seeds are depicted in Figure 4 showing that the naïve generalization approach has a similar performance with policies



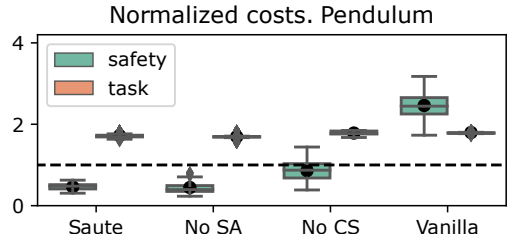
(a) Safety costs

(b) Task costs

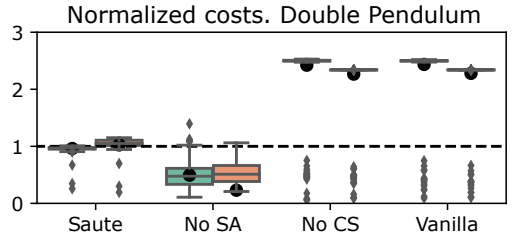
Figure 4. The normalized task and safety costs for generalization across safety budgets. The task costs are divided by $-2d$, while the safety costs by d with d being the safety budget, hence dashed lines indicate the safety threshold. The *baseline* policies are trained and evaluated on the same safety budges; the *naïve* approach trains policies on the safety budget 60 and the *zero-shot* approach trains policy by sampling the safety budget from the interval $[5, 100]$. For task costs the higher values are better.

explicitly trained for budgets 40 and 80. We further train another set of policies with the initial safety state uniformly sampled from the interval $[5, 100]$. When it comes to safety constraint satisfaction this *zero-shot* approach outperforms the naïve approach, which has some outlier trajectories for safety budgets 40 and 80 (see Figure 4).

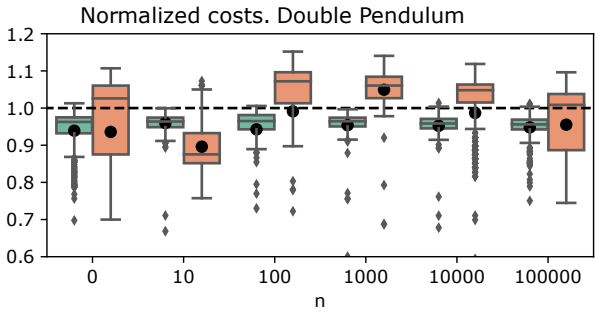
Ablation. Since we have only two main components (cost shaping and state augmentation) our ablation study is rather straightforward. We perform evaluations on the double pendulum environment with the safety budget set to 40. According to the results in Figure 5(b) removing cost shaping produce results similar to Vanilla TRPO, which is expected. Removing state augmentation leads to a significant deterioration in task cost minimization and the safety costs are much lower. It appears that the trained policy hedges its bets by not using the whole safety budget, which leads to the task cost increase. Interestingly, in the pendulum swing-up case, removing the state augmentation is not catastrophic, see Figure 5(a). This is because we evaluate the policy while sampling the initial states near the same initial position. Hence the safety states are similar at every time for different trajectories. This allows the algorithm without



(a) Ablation on components. Pendulum Swing-up

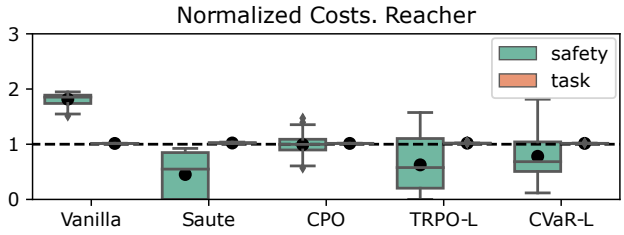


(b) Ablation on components. Double Pendulum

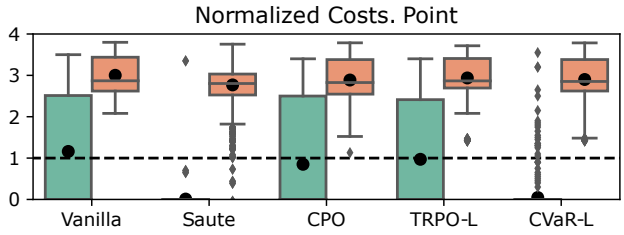


(c) Task cost shaping

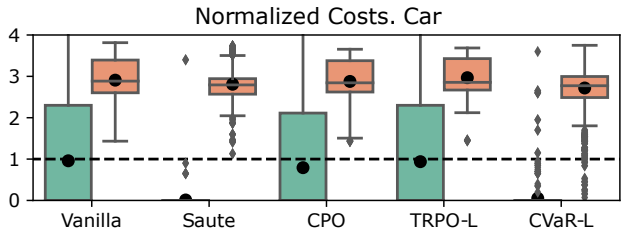
Figure 5. Normalized safety (on the left) and task (on the right) costs in ablation studies for Sauté TRPO. Panels a and b: ablation on components for double pendulum (Panel a) and pendulum swing-up (Panel b). "No SA" stands for no state augmentation, "No CS" stands for no cost shaping. Panel c: varying values n in reshaped costs $\widetilde{c}_n$ for the safety budget $d = 40$. The numerical values can be found in Table A3 in Appendix. In all plots the task costs are divided by -80 (i.e., higher values are better), while the safety costs by 40, dashed lines indicate the safety threshold.



(a) Reacher



(b) Safety gym. Point



(c) Safety gym. Car

Figure 6. Normalized safety (on the left) and task (on the right) costs in Reacher and Safety Gym. Panel a: Results for the reacher environment with the safety budget 10, where the task costs are divided by 60, while the safety costs by 10. Panel b: Results for the safety gym point environment with the safety budget 20, where the task costs are divided by -1 and the safety costs are divided by 20. Dashed lines indicate the safety threshold. For task costs the higher values are better.

state augmentation to still produce competitive results. This experiment shows that the effect of state augmentation may be easily overlooked.

All the parameters in a "sautéed" algorithm are the same as in its "vanilla" version except for the parameter n in the cost $\widetilde{c}_n$. Hence we only need to perform the second ablation with respect to n. In all our previous experiments with the double pendulum, we set $n = 200$, and here we test different values n for the safety budget 40 and present the evaluation results after 600 epochs of training in Figure 5(c). Increasing n from 0 to 100 improves the performance of Sauté RL by decreasing the number of outliers in safety cost distributions. However, increasing the value of n to 10000 and 100000 leads to additional outliers in safety and task cost distributions. We attribute this to numerical issues as the task costs c take values between zero and one, and the large value of n can lead to numerical issues in training. We note, however, that the differences between the numerical values are not large, which can be verified in Table A3 in Appendix.

Further Experiments. We then test Sauté TRPO on Reacher Environment as well as a Safety Gym environment with Car and Point robots. Results in Figure 6(a) are quite similar to the results on the pendulum swing-up and double pendulum environments, where both TRPO Lagrangian (TRPO-L) and CPO delivered the policies safe on average, but not safe almost surely as Sauté TRPO. We also compared our implementation of the CVaR constrained problem with $\alpha = 0.01$, which is denoted as CVaR-L and is an approximation of almost surely safety constraints (see Appendix C.2). While the reacher task is closer in nature

to pendulums, the safety gym is quite a complicated environment. Every environment in the safety gym has dozens of states (46 for the point environment and 56 for the car environment) due to the inclusion of LIDAR measurements. Finally, the instantaneous task costs are shaped so that their values are close to zero, which is tailored for TRPO and PPO-like algorithms. This makes our approach a bit harder to use since we reshape the costs. Nevertheless, the results in Figure 6(b) indicate that Sauté RL delivers a safe set of policies with only a few outliers violating the constraints. Most of the trajectories have the same returns as TRPO Lagrangian, but the average task cost is brought down by a few outliers. It appears that in these outlier trajectories the "sautéed" policies prioritize safety over task costs. While TRPO-L and CPO produce on average better task costs, the safety constraints are being violated on a rather regular basis. Note that in our experiments the safety budget is chosen to be the average incurred cost by Vanilla TRPO and neither TRPO-L nor CPO decreases the variance of the safety cost. While TRPO-L and CPO lower the average cost, the number of outlier trajectories remains quite high. Finally, CVaR-L fails for the Reacher environment but appears to be a somewhat competitive approximation for Sauté RL for the Point and Car environments. Our results suggest, however, that approximating almost surely constraints with CVaR constraints can still result in a significant number of outliers.

We further compared our method to a more advanced baseline where a PID controller is used to update the Lagrangian multiplier (Stooke et al., 2020). Our results suggest that PID Lagrangian offers some level of improvement in training (especially stability of the training process). However, the end performance is similar to TRPO-L in our environments (see results in Appendix F.2). This is because we consider a completely different problem formulation to PID Lagrangian and Lagrangian approaches.

5. Conclusion

We presented an approach to safe RL using state augmentation, which we dubbed Sauté RL. The key difference of our approach is that we ensure the safety constraints almost surely (with probability one), which is desirable in many applications. Even, in deterministic environments having an "average" constraint can lead to unwanted effects, when the safety cost is high for some initial states and is low for other initial states at the same time. Our approach deals with this case by ensuring that the same constraint is satisfied for all initial states. We showed that state augmentation is essential in some environments for optimality, which supports our theoretical results implying that the optimal policy depends on the safety state. The constraint satisfaction almost surely is a very strong criterion and in some applications, it can be too restrictive. This is, however, a design choice and application dependent, now let us discuss specific pros and cons.

Advantages. Sauté RL has a plug-n-play nature, which allows for straightforward extensions as we demonstrated by sautéing PPO, TRPO, and SAC. Furthermore, we used sautéed environments in the model-based RL setting (MBPO and PETS) as well. We showed that Sauté RL generalizes across safety budgets and can learn safe policies for all safety budgets simultaneously. This feature is enabled by the architecture of our state augmentation. Since the remaining safety budget is the initial state, we can randomly sample different safety budgets at the beginning of the episode. At the test time, the safety budget could be set in a deterministic fashion to evaluate specific Safe RL problems.

Limitations. We have not treated the case with multiple constraints, which can bring some difficulties for cost reshaping. Further, "sautéing" an MDP increases the state-space by the number of constraints. Therefore, the dimension of value functions and policy grows and potentially can lead to scalability issues. While this is a common issue in constrained problems, using a Lagrangian approach can be more sample efficient. Since the theoretical sample efficiency estimates usually depend on the number of states (Mania et al., 2018), it would be interesting to find means to counteract this loss of efficiency. This potentially can be done in the setting where the safety cost function is given, by exploiting the known safety state transitions.

Sauté RL does not currently address the problem of constraint violation during training, which is still a major problem in safe RL. However, the combination of Sauté RL and methods for addressing such a problem could be an interesting direction for future work as well. After all, the cost-to-go in Sauté RL does incorporate potential safety violations and this information can potentially be used for safe training.

Future Work. Besides addressing the limitations, it would be interesting to extend our approach to model-based algorithms applicable to high-dimensional environments including PlaNet (Hafner et al., 2019), Dreamer (Hafner et al., 2019), Stochastic Latent Actor Critic (Lee et al., 2020) etc. Furthermore, it would be interesting to evaluate the effect of safety state augmentation on average and CVaR constrained problems.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31, 2017.

Akametalu, A. K., Fisac, J. F., Gillula, J. H., Kaynama, S., Zeilinger, M. N., and Tomlin, C. J. Reachability-based safe learning with Gaussian processes. In *IEEE Conference on Decision and Control*, pp. 1424–1431, 2014.

Altman, E. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Arapostathis, A., Borkar, V. S., Fernández-Gaucherand, E., Ghosh, M. K., and Marcus, S. I. Discrete-time controlled Markov processes with average cost criterion: a survey. *SIAM Journal on Control and Optimization*, 31(2):282–344, 1993.

Bäuerle, N. and Ott, J. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3):361–379, 2011.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, pp. 908–918, 2017.

Bertsekas, D. P. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Bharadhwaj, H., Kumar, A., Rhinehart, N., Levine, S., Shkurti, F., and Garg, A. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Calvo-Fullana, M., Paternain, S., Chamon, L. F., and Ribeiro, A. State augmented constrained reinforcement learning: Overcoming the limitations of learning with rewards. *arXiv preprint arXiv:2102.11941*, 2021.

Cheng, R., Orosz, G., Murray, R. M., and Burdick, J. W. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. *arXiv preprint arXiv:1903.08792*, 2019.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A Lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 8092–8101, 2018.

Chow, Y., Nachum, O., Faust, A., Ghavamzadeh, M., and Duenez-Guzman, E. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

Cowen-Rivers, A. I., Palenicek, D., Moens, V., Abdullah, M. A., Sootla, A., Wang, J., and Bou-Ammar, H. SAMBA: Safe model-based & active reinforcement learning. *Machine Learning*, pp. 1–31, 2022.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Daryin, A. and Kurzhanski, A. Nonlinear control synthesis under double constraints. *IFAC Proceedings Volumes*, 38 (1):247–252, 2005.

Dean, S., Tu, S., Matni, N., and Recht, B. Safely learning to control the constrained linear quadratic regulator. In *American Control Conference*, pp. 5582–5588, 2019.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pp. 465–472, 2011.

Ding, D., Zhang, K., Basar, T., and Jovanovic, M. R. Natural policy gradient primal-dual method for constrained Markov decision processes. In *NeurIPS*, 2020.

Eysenbach, B., Gu, S., Ibarz, J., and Levine, S. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018.

Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., and Tomlin, C. J. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7): 2737–2752, 2019. doi: 10.1109/TAC.2018.2876389.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

Hernández-Lerma, O. and Lasserre, J. B. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media, 2012.

Hernández-Lerma, O. and Muñoz de Ozak, M. Discrete-time Markov control processes with discounted unbounded costs: optimality criteria. *Kybernetika*, 28(3): 191–212, 1992.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, pp. 12519–12530, 2019.

Kamthe, S. and Deisenroth, M. Data-efficient reinforcement learning with probabilistic model predictive control. In *International Conference on Artificial Intelligence and Statistics*, pp. 1701–1710. PMLR, 2018.

Koller, T., Berkenkamp, F., Turchetta, M., and Krause, A. Learning-based model predictive control for safe exploration. In *IEEE Conference on Decision and Control*, pp. 6059–6066, 2018.

Lee, A., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33, 2020.

Mania, H., Guy, A., and Recht, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Mguni, D., Jennings, J., Jafferjee, T., Sootla, A., Yang, Y., Yu, C., Islam, U., Wang, Z., and Wang, J. DESTA: A framework for safe reinforcement learning with markov games of intervention. *arXiv preprint arXiv:2110.14468*, 2021.

Ohnishi, M., Wang, L., Notomista, G., and Egerstedt, M. Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Transactions on Robotics*, 35(5):1186–1205, 2019.

Ott, J. T. A Markov decision model for a surveillance application and risk-sensitive Markov decision processes, 2010. PhD Thesis.

Pineda, L., Amos, B., Zhang, A., Lambert, N. O., and Calandra, R. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL `https://arxiv.org/abs/2104.10159`.

Polymenakos, K., Rontsis, N., Abate, A., and Roberts, S. SafePILCO: A software tool for safe and data-efficient policy synthesis. In Gribaudo, M., Jansen, D. N., and Remke, A. (eds.), *Quantitative Evaluation of Systems*, pp. 18–26. Springer International Publishing, 2020.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Ray, A., Achiam, J., and Amodei, D. Benchmarking safe exploration in deep reinforcement learning, 2019. URL `https://cdn.openai.com/safexp-short.pdf`.

Rockafellar, R. T., Uryasev, S., et al. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sootla, A., Cowen-Rivers, A. I., Jafferjee, T., and Wang, Z. Sauté RL: Almost surely safe reinforcement learning using state augmentation, 2022. URL `https://github.com/huawei-noah/HEBO/tree/master/SAUTE`.

Stooke, A., Achiam, J., and Abbeel, P. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tamar, A., Glassner, Y., and Mannor, S. Optimizing the cvar via sampling. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Turchetta, M., Berkenkamp, F., and Krause, A. Safe exploration in finite Markov decision processes with Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 4312–4320, 2016.

Turchetta, M., Kolobov, A., Shah, S., Krause, A., and Agarwal, A. Safe reinforcement learning via curriculum induction. *arXiv preprint arXiv:2006.12136*, 2020.

Wachi, A., Sui, Y., Yue, Y., and Ono, M. Safe exploration and optimization of constrained MDPs using Gaussian processes. In *AAAI Conference on Artificial Intelligence*, 2018.

Waskom, M. L. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. URL `https://doi.org/10.21105/joss.03021`.

Yang, Q., Simão, T. D., Tindemans, S. H., and Spaan, M. T. WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning. In *AAAI Conference on Artificial Intelligence.*, 2021.

Yang, T.-Y., Rosca, J., Narasimhan, K., and Ramadge, P. J. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2019.

Zimmer, C., Meister, M., and Nguyen-Tuong, D. Safe active learning for time-series modeling with Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 2730–2739, 2018.

A. Theoretical analysis

Proof of Theorem 2 follows form the results by (Hernández-Lerma & Muñoz de Ozak, 1992) and (Hernández-Lerma & Lasserre, 2012), which we reproduce and condense for readers' convenience. We cover the conditions for the existence of the Bellman equation and optimal policies in Appendix A.1, and we discuss the convergence of a sequence of MDPs to a limit MDP in Appendix A.2 and discuss the application of these results to our case in Appendix A.3.

A.1. MDPs, Optimality and Bellman equation

Consider an MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, c, \gamma_c\}$ with an action set defined for every state $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s})$, where $\mathcal{A}$ are non-empty sets. The set

$$\mathbb{K} = \{(\boldsymbol{s}, \boldsymbol{a}) | \boldsymbol{s} \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}(\boldsymbol{s})\}$$

of admissible state-action pairs is assumed to be a Borel subset of $\mathcal{S} \times \mathcal{A}$. We will need the following definitions:

Definition A1 *A function u is* inf-compact *on $\mathbb{K}$ if the set $\{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s}) | u(\boldsymbol{s}, \boldsymbol{a}) \leq r\}$ is compact for every $\boldsymbol{s} \in \mathcal{S}$ and $r \in \mathbb{R}$.*

A function u is lower semi-continuous (l.s.c.) *in $\mathcal{S}$ if for every $\boldsymbol{s}_0 \in \mathcal{S}$ we have $\liminf_{\boldsymbol{s} \to \boldsymbol{s}_0} u(\boldsymbol{s}) \geq \boldsymbol{s}_0$.*

A set-valued function $\boldsymbol{s} \to \mathcal{A}(\boldsymbol{s})$ is lower semi-continuous (l.s.c.)*, if for any $\boldsymbol{s}_n \to \boldsymbol{s}$ in $\mathcal{S}$ and $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s})$, there are $\boldsymbol{a}_n \in \mathcal{A}(\boldsymbol{s}_n)$ such that $\boldsymbol{a}_n \to \boldsymbol{a}$.*

A distribution $\mathcal{Q}(\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{a})$ is called weakly continuous*, if for any continuous and bounded function u on $\mathcal{S}$ the map $(\boldsymbol{s}, \boldsymbol{a}) \to \int_{\mathcal{S}} u(\boldsymbol{y}) \mathcal{Q}(d\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{a})$ is continuous on $\mathbb{K}$.*

Let the value functions be denoted as follows:

$$V(\pi, \boldsymbol{s}_0) = \mathbb{E}_{\boldsymbol{s}}^{\pi} \sum_{t=0}^{\infty} \gamma_c^t c(\boldsymbol{s}_t, \boldsymbol{a}_t),$$
$$V^*(\boldsymbol{s}) \triangleq \inf_{\pi} V(\pi, \boldsymbol{s}),$$

where $\mathbb{E}_{\boldsymbol{s}}^{\pi}$ stands for the average with action sampled according to the policy π and the transitions $\mathcal{P}$. We also define the Bellman operator:

$$Tv(\boldsymbol{s}) = \min_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s})} \left[c(\boldsymbol{s}, \boldsymbol{a}) + \gamma \int v(\boldsymbol{y}) \mathcal{P}(d\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{a}) \right],$$

acting on value functions. We also make the following assumptions:

B1. The function $c(\boldsymbol{s}, \boldsymbol{a})$ is bounded, measurable on $\mathbb{K}$, nonnegative, lower semi-continuous and inf-compact on $\mathbb{K}$;

B2. The transition law $\mathcal{P}$ is weakly continuous;

B3. The set valued map $\boldsymbol{s} \to \mathcal{A}(\boldsymbol{s})$ is lower semi-continuous;

We summarize Theorems 4.2 and 4.6 by (Hernández-Lerma & Muñoz de Ozak, 1992) in the following result:

Proposition A1 *Suppose an MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, c, \gamma_c\}$ satisfies Assumptions B1-B3. Then:*

a) The optimal cost function V^* satisfies the Bellman equation, i.e., $TV^* = V^*$ (Theorem 4.2);

b) The policy π^* is optimal (i.e., $V(\pi^*, \cdot) = V^*(\cdot)$) if and only if $V(\pi^*, \cdot) = TV(\pi^*, \cdot)$ (Theorems 4.2 and 4.6).

(Hernández-Lerma & Muñoz de Ozak, 1992) proved these results under milder conditions on the cost function than the boundedness condition we use. However, (Hernández-Lerma & Muñoz de Ozak, 1992) also had an assumption on the existence of a feasible policy, i.e., they assumed that there exists a policy $\widehat{\pi}$ such that $V(\widehat{\pi}, \boldsymbol{s}) < \infty$ for each $\boldsymbol{s} \in \mathcal{S}$. This, however, follows from the boundedness of the cost function.

A.2. Limit of a sequence of MDPs

Consider now a sequence of MDPs $\mathcal{M}_n = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, c_n, \gamma_c\}$, where without loss of generality we will write $c \triangleq c_\infty$ and $\mathcal{M} \triangleq \mathcal{M}_\infty$. Consider now a sequence of value functions $\{V_n^*\}_{n=0}^{\infty}$:

$$V_n(\pi, \boldsymbol{s}_0) = \mathbb{E}_{\boldsymbol{s}}^{\pi} \sum_{t=0}^{\infty} \gamma_c^t c_n(\boldsymbol{s}_t, \boldsymbol{a}_t),$$
$$V_n^*(\boldsymbol{s}) \triangleq \inf_{\pi} V_n(\pi, \boldsymbol{s}).$$

The "limit" value functions (with $n = \infty$) we still denote as follows:

$$V(\pi, \boldsymbol{s}_0) = \mathbb{E}_{\boldsymbol{s}}^{\pi} \sum_{t=0}^{\infty} \gamma_c^t c(\boldsymbol{s}_t, \boldsymbol{a}_t),$$
$$V^*(\boldsymbol{s}) \triangleq \inf_{\pi} V(\pi, \boldsymbol{s}).$$

We also define the sequence of Bellman operators

$$T_n v(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\mathcal{A}(\boldsymbol{s})} \left[c_n(\boldsymbol{s},\boldsymbol{a}) + \gamma \int v(\boldsymbol{y})\mathcal{P}(d\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) \right],$$
$$T v(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\mathcal{A}(\boldsymbol{s})} \left[c(\boldsymbol{s},\boldsymbol{a}) + \gamma \int v(\boldsymbol{y})\mathcal{P}(d\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) \right].$$

In addition to the previous assumptions, we make an additional one, while modifying Assumption B1:

B1'. For each n the functions $c_n(\boldsymbol{s},\boldsymbol{a})$ are bounded, measurable on $\mathbb{K}$, nonnegative, lower semi-continuous and inf-compact on $\mathbb{K}$;

B4. The sequence $\{c_n(\boldsymbol{s},\boldsymbol{a})\}_{n=0}^{\infty}$ is such that $c^n \uparrow c$;

We reproduce Theorem 5.1 by (Hernández-Lerma & Muñoz de Ozak, 1992) in the following proposition:

Proposition A2 *Suppose MDPs $\mathcal{M}_n = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, c_n, \gamma_c\}$ satisfy Assumptions B1', B2 - B4, then the sequence $\{V_n^*\}$ is monotonically increasing and converges to V^*.*

Note that we do not require the cost function c to be bounded. This, however, comes at a price in that we cannot generally claim that $TV^* = V^*$. To ensure this property we need additional assumptions (see (Hernández-Lerma & Muñoz de Ozak, 1992)).

A.3. Proof of Theorem 2

Coming back to the Sauté MDP, recall that we define the following cost function for $\widetilde{\mathcal{M}}_n$:

$$\widetilde{c}_n(\boldsymbol{s}_t,\boldsymbol{z}_t,\boldsymbol{a}_t) = \begin{cases} c(\boldsymbol{s}_t,\boldsymbol{a}_t) & \boldsymbol{z}_t \geq 0, \\ n & \boldsymbol{z}_t < 0. \end{cases}$$

Therefore, to prove Theorem 2 we need to verify that Sauté MDP $\widetilde{\mathcal{M}}_n$ satisfying Assumptions A1-A3 also satisfies Assumptions B1', B2-B4. According to Assumptions A1-A2, we consider bounded, continuous costs c with compact action space $\mathcal{A}$, hence Assumptions B1', B3, and B4 are satisfied. Assumptions B2 and A3 are identical. Note that, if the transition function $\mathcal{P}$ is a Gaussian with continuous mean and variance, then Assumption A3 is satisfied (cf. (Arapostathis et al., 1993)).

B. State augmentation techniques

B.1. By Daryin and Kurzhanski

(Daryin & Kurzhanski, 2005) considered the classical control problem, i.e., the model is assumed to be known and the goal is to compute the optimal policy. They consider the following model:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) + \boldsymbol{C}(t)\boldsymbol{v}(t),$$

where $\boldsymbol{x}(t)$ is the state, $\boldsymbol{u}(t)$ is the control signal and $\boldsymbol{v}(t)$ is an unknown disturbance. Both controls and disturbances are subject to hard bounds:

$$\boldsymbol{u}(t) \in \mathcal{U}(t), \qquad \boldsymbol{v}(t) \in \mathcal{V}(t),$$

where the time-varying sets $\mathcal{U}(t)$ and $\mathcal{V}(t)$ are also known. The controls are also subject to soft constraints:

$$\int_{t_0}^{t_1} \|\boldsymbol{u}(t)\|^2_{\boldsymbol{R}(t)} dt \leq \boldsymbol{k}(t_0).$$

To avoid dealing with two-types of constraints the authors proposed to augment the state-space with a new state $k(t)$ as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) + \boldsymbol{C}(t)\boldsymbol{v}(t),$$
$$\dot{\boldsymbol{k}} = -\|\boldsymbol{u}(t)\|^2_{\boldsymbol{R}(t)}.$$

Now the integral constraint can be enforced as an end point constraint $k(t_1) \geq 0$. We will not go into further detail about this work but mention that all the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{R}$ as well as set-valued maps $\mathcal{U}(t)$, $\mathcal{V}(t)$ are assumed to be known. In our case, we assume unknown dynamics.

B.2. By Chow et al

(Chow et al., 2017) considered safe RL with CVaR constraints and addressed the following optimization problem:

$$\min_{\pi,\nu} \mathbb{E} \sum_{t=0}^{T} \gamma_c^t c(\boldsymbol{s}_t,\boldsymbol{a}_t), \tag{A1}$$
$$\text{s.t.: } \boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t,\boldsymbol{s}_{t-1},\boldsymbol{a}_{t-1},\ldots,,\boldsymbol{s}_0,\boldsymbol{a}_0)$$
$$\nu + \frac{1}{1-\alpha}\mathbb{E}\,\text{ReLU}\left(\sum_{t=0}^{T}\gamma_l^t l(\boldsymbol{s}_t,\boldsymbol{a}_t) - \nu\right) \leq d,$$

where T is the control horizon. In this formulation, one also needs to consider the target state $\boldsymbol{s}_{\text{Tar}}$, which signifies the end of the episode.

For their state augmentation approach (Chow et al., 2017) proposed the following augmented MDP:

$$\begin{aligned} &\boldsymbol{s}_{t+1} \sim p(\cdot|\boldsymbol{s}_t,\boldsymbol{a}_t), \boldsymbol{s}_0 \sim \mathcal{S}_0, \\ &\boldsymbol{x}_{t+1} = (\boldsymbol{x}_t - l(\boldsymbol{s}_t,\boldsymbol{a}_t))/\gamma_l, \boldsymbol{x}_0 = \nu, \end{aligned} \tag{A2}$$

and the augmented cost

$$\widetilde{c}_\lambda(\boldsymbol{s}_t,\boldsymbol{x}_t,\boldsymbol{a}_t) = \begin{cases} c(\boldsymbol{s}_t,\boldsymbol{a}_t) & \boldsymbol{s}_t \neq \boldsymbol{s}_{\text{Tar}} \\ \dfrac{\lambda\text{ReLU}(-\boldsymbol{x}_t)}{1-\alpha} & \text{otherwise} \end{cases} \tag{A3}$$

The optimization problem that they considered was as follows:

$$\min_{\pi,\nu}\max_{\lambda\geq 0} \mathbb{E}\sum_{t=0}^{T}\gamma_c^t \widetilde{c}_\lambda(\boldsymbol{s}_t,\boldsymbol{x}_t,\boldsymbol{a}_t), \tag{A4}$$
$$\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t,\boldsymbol{x}_t).$$

This idea of state augmentation was introduced by (Ott, 2010; Bäuerle & Ott, 2011) who showed that the optimal policy of a CVaR optimization problem (unconstrained) must depend on the current state $\boldsymbol{s}_t$ as well as the history of the accumulated cost $\boldsymbol{x}_t$. In our understanding, however, the representation of the optimal policy for the CVaR-constrained optimization was not discussed by (Ott, 2010; Bäuerle & Ott, 2011). Hence there may still be an open question of the validity of the Bellman equation for Equation A1 and the representation of the optimal policy. We stress that (Chow et al., 2017) provided many other theoretical results justifying their approach to CVaR constrained reinforcement learning, but perhaps some gaps remain.

Note that the finite horizon is only a technical difference between our formulation and the formulation by (Chow et al., 2017) and is of no consequence. The difference is the initial value of the augmented state $\boldsymbol{x}_0$, which is equal to ν instead of d as it is in our case. Although this seems to be a subtle difference, it allows for many features such as plug-n-play methods, generalization across safety budgets, learning safe policy for all safety budgets d in some interval $[d_{\text{lower}}, d_{\text{upper}}]$.

B.3. By Calvo-Fullana et al

The authors consider the following problem

$$\begin{aligned} &\max_{\pi} \lim_{T\to\infty} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\pi} \sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t), \\ &\text{s.t.: } \lim_{T\to\infty} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\pi} \sum_{t=0}^{T} r_i(\boldsymbol{s}_t, \boldsymbol{a}_t) \geq d_i, \\ &\qquad \boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \dots, , \boldsymbol{s}_0, \boldsymbol{a}_0), \end{aligned} \tag{A5}$$

and denoted the objective as

$$V_i(\pi) \triangleq \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\pi} \left[\sum_{t=0}^{T} r_i(\boldsymbol{s}_t, \boldsymbol{a}_t) \right],$$

and the optimal cost (sic) as $V_0(\pi^*)$. Then the authors defined the Lagrangian for the primal-dual solution:

$$\mathcal{L}(\pi, \lambda) = V_0(\pi) + \sum_{i=1}^{m} \lambda_i \left(V_i(\pi) - c_i \right).$$

The solution was proposed by computing $\underset{\pi}{\operatorname{argmax}}\, \mathcal{L}(\pi, \lambda)$, where the optimal Lagrangian multipliers need to be optimized over. (Calvo-Fullana et al., 2021) propose to update the multipliers as follows:

$$\lambda_{i,k+1} = \left[\lambda_{i,k} - \frac{\eta_\lambda}{T_0} \sum_{t=kT_0}^{(k+1)T_0 - 1} \left(r_i(\boldsymbol{s}_t, \boldsymbol{a}_t) - d_i \right) \right],$$

where η_λ is the step size, T_0 is the epoch duration, k is the iteration index. The policy π in this formulation depends on the state $\boldsymbol{s}_t$ and Lagrangian multipliers λ_i. Using this idea the authors show that there exists a policy that allows constraint satisfaction with probability one:

$$\lim_{T\to\infty} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\pi} \sum_{t=0}^{T} r_i(\boldsymbol{s}_t, \boldsymbol{a}_t) \geq d_i, \forall i \text{ a. s.}$$

similarly to our case. As we discuss, however, it is not necessary to use the Lagrangian formulation and such complicated constructions to arrive at a similar conclusion. Furthermore, it is not clear if the algorithm trains a policy satisfying the constraint almost surely.

C. Implementation Details

Our implementation is freely available online (Sootla et al., 2022).

C.1. Sauté RL

The main benefit of our approach to safe RL is the ability to extend it to *any critic-based RL algorithm*. This is because we do not need to change the algorithm itself (besides some cosmetic changes), but create a wrapper around the environment. The implementation is quite straightforward and the only "trick" we had to resort to is normalizing the safety state by dividing with the safety budget:

$$\boldsymbol{z}_{t+1} = (\boldsymbol{z}_t - l(\boldsymbol{s}_t, \boldsymbol{a}_t)/d)/\gamma_l, \boldsymbol{z}_0 = 1.$$

Hence the variable $\boldsymbol{z}_t$ is always between zero and one.

```
def safety_step(self, cost: np.ndarray) -> np.ndarray:
    """
    Update the normalized safety state
    """
    # subtract the normalized cost
    self._safe_state -= cost / self.safe_budget
    # normalize by the discount factor
    self._safe_state /= self.safe_discount_factor
    return self._safe_state
```

The step function has to be overloaded to augment the safe state and shape the cost.

```
def step(self, action:np.ndarray) -> np.ndarray:
    """
    Step into the environment
    """
    # get the state of the environment
    next_obs, reward, done, info = super().step(action)
    # get the safe state
    next_safe_state = self.safety_step(info['cost'])
    # shape the reward
    if next_safe_state <= 0:
        reward = 0
    # augment the state
    augmented_state = np.hstack([next_obs,
                                 next_safe_state])
    # save values
    info['true_reward'] = reward
    info['next_safe_state'] = next_safe_state
    return augmented_state, reward, done, info
```

Note that in this implementation, we assume that the minimum reward is zero and the maximum reward is nonnegative. In some environments, we set a different minimum reward (i.e., we shape $\widetilde{c}_n$ with a different value n). Finally, resetting the environment requires only state augmentation.

```
def reset(self) -> np.ndarray:
    """
    Reset the environment
    """
    # get the state of the environment
    state = super().reset()
    # reset the safe state
    self._safe_state = 1.0
    # augment the state
    augmented_state = np.hstack([state,
                                 self._safe_state])
    return augmented_state
```

We used safety starter agents (Ray et al., 2019) (Tensorflow == 1.13.1) as the core implementation for model-free methods and their Lagrangian versions (PPO, TRPO, SAC, CPO). We have tested some environments using stable baselines library (Raffin et al., 2021) (PyTorch >= 1.8.1) and did not find drastic performance differences. Our choice of safety starter agents is guided only by the implementation of the Lagrangian version of PPO, TRPO, and SAC as well as CPO, which we modify by sautéing PPO, TPRO, and SAC.

C.2. CVaR Lagrangian

Our implementation of CVaR Lagrangian leverages the implementation by (Cowen-Rivers et al., 2022) while implementing the algorithm within the safety starter agents (Ray et al., 2019). Our implementation is quite straightforward as all we need to do is to replace the empirical estimator of the mean with the empirical CVaR estimator. First, let us recall an alternative definition of CVaR for a random variable X with a probability density function $p(X)$ and cumulative probability distribution $F_X(x) = \mathbb{P}(X < x) = \int_{X<x} p(X)dX$ given by (Rockafellar et al., 2000):

$$\text{VaR}_\alpha(X) = \min\{x \in \mathbb{R} | F_X(x) \geq \alpha\},$$
$$\text{CVaR}_\alpha(X) = (1-\alpha)^{-1} \int\limits_{F_X(x) \geq \text{VaR}_\alpha(X)} F_X(x)p(X)dX.$$

This definition is equivalent to ours as shown in (Rockafellar et al., 2000). Note that this definition implies that $\text{CVaR}_\alpha(X)$ is the mean of the tail of the distribution defined by quantile α. Therefore, instead of using the definition

$$\text{CVaR}_\alpha(X) = \min_{\nu \in \mathbb{R}} \left(\eta + \frac{1}{1-\alpha} \mathbb{E}\, \text{ReLU}(X - \nu) \right)$$

we can simply compute an empirical estimate of the quantile and then compute the empirical estimate of the mean of the tail. Following (Tamar et al., 2015) our algorithm for the empirical CVaR estimate is as follows.

1. Let $x_1, \ldots, x_n$ be drawn from X and let $\alpha \in (0, 1)$;
2. Determine the quantile $q = \lceil n(1-\alpha) \rceil$;
3. Sort the samples to get $x_{i_1}, \ldots, x_{i_n}$ with $x_{i_1} < \cdots < x_{i_n}$ and $i_1, \ldots, i_n$ being a permutation of $1, \ldots, n$;
4. Compute the empirical estimate as follows:

$$\widehat{\text{CVaR}_\alpha}(X) = \frac{1}{n-q} \sum_{j=q}^{n} x_{i_j}.$$

Now this estimate can simply replace the empirical mean estimate for the computation of advantage for the safety cost. No other changes to the algorithms are required.

D. Environments

Pendulum Swing-up. We take the single pendulum swing-up from the classic control library in the Open AI Gym (Brockman et al., 2016). However, we define the instantaneous task cost following (Cowen-Rivers et al., 2022) as follows:

$$c(\boldsymbol{s}, \boldsymbol{a}) = 1 - \frac{\boldsymbol{\theta}^2 + 0.1\dot{\boldsymbol{\theta}}^2 + 0.001\boldsymbol{a}^2}{\pi^2 + 6.404},$$

which takes values between zero and one, since $\boldsymbol{a} \in [-2, 2]$, $\boldsymbol{s}[0] = \boldsymbol{\theta} \in [-\pi, \pi]$ $\boldsymbol{s}[1] = \dot{\boldsymbol{\theta}} \in [-8, 8]$. We define the instantaneous safety cost following (Cowen-Rivers et al., 2022):

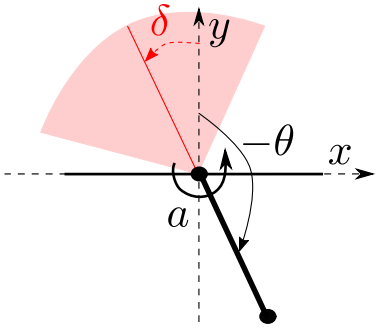
$$l = \begin{cases} 1 - \dfrac{|\theta - \delta|}{50} & \text{if } -25 \leq \theta \leq 75, \\ 0 & \text{otherwise,} \end{cases}$$

where θ is the angle (in degrees) of the pole deviation from the upright position. The cost is designed to create a trade-off between swinging up the pendulum and keeping away from the angle $\delta = 25^o$. This environment has three states (cosine and sine of θ, as well as angular velocity $\dot{\theta}$) and one action. See the depiction in Figure A1(a).

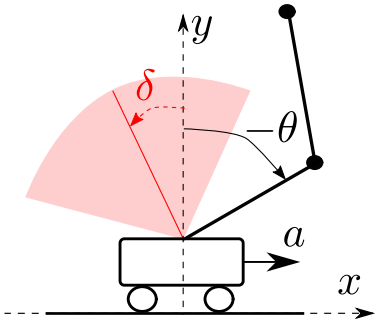
Double Pendulum. We take the double pendulum stabilization implementation by (Todorov et al., 2012) using the Open AI Gym (Brockman et al., 2016) interface (the environment `InvertedDoublePendulumEnv` from `gym.envs.mujoco`). We modify the environment by setting the maximum episode length to 200 and divide the instantaneous reward by 10. We used $n = 200$ to get $\widetilde{c}_n$.

We define safety similarly to the pendulum swing-up case, i.e., we use the same instantaneous cost with θ is angle (in degrees) of the first pole deviation from the upright position and define the cost as follows:

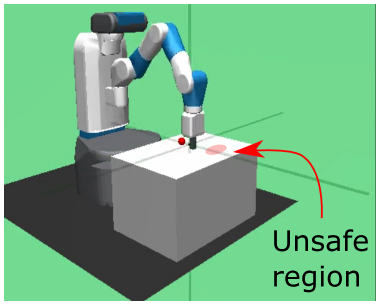
$$l = \begin{cases} 1 - \dfrac{|\theta - \delta|}{50} & \text{if } -25 \leq \theta \leq 75, \\ 0 & \text{otherwise.} \end{cases}$$



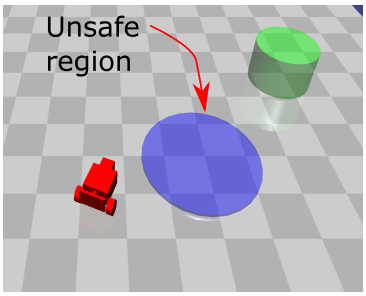
(a) Safe pendulum swing-up.



(b) Safe double pendulum



(c) Safe reacher



(d) Safety gym

Figure A1. Panels a and b: safe pendulum environments. In both cases, θ - is the angle from the upright position, a is the action, δ - is the unsafe pendulum angle, the safety cost is the distance toward the unsafe pendulum angle, which is incurred only in the red area. Panel e: safe reacher: the robot needs to avoid the unsafe region. Panel d: a schematic depiction of the safety gym environment: robot needs to reach the goal while avoiding the unsafe region.

Table A1. Default hyperparameters for TRPO, PPO, CPO

	Name	Value
Common parameters	Network architecture	[64,64]
	Activation	tahn
	Value function learning rate	1e-3
	Task Discount Factor	0.99
	Lambda	0.97
	N samples per epochs	1000
	N gradient steps	80
	Target KL	0.01
Safety	Penalty learning rate	5e-2
	Safety Discount Factor	0.99
	Safety Lambda	0.97
	Initial penalty	1
PPO	Clip ratio	0.2
	Policy learning rate	3e-4
	Policy iterations	80
	KL margin	1.2
TRPO/CPO	Damping Coefficient	0.1
	Backtrack Coefficient	0.8
	Backtrack iterations	10
	Learning Margin	False

This environment has eleven states and one action. See the depiction in Figure A1(b).

Reacher. We take the reacher implementation by (Todorov et al., 2012) using the Open AI Gym (Brockman et al., 2016) interface (`Reacher`). We add the following safety cost for this environment

$$l = \begin{cases} 100 - 50 \cdot |\boldsymbol{x} - \boldsymbol{x}_{\text{target}}| & \text{if } |\boldsymbol{x} - \boldsymbol{x}_{\text{target}}| \leq 0.5, \\ 0 & \text{otherwise,} \end{cases}$$

where $\boldsymbol{x}_{\text{target}}$ is the position of the target (set to $\begin{pmatrix}1.0 & 1.0 & 0.01\end{pmatrix}$) and $\boldsymbol{x}$ is the position of the arm in Cartesian coordinates. The environment is schematically depicted in Figure A1(c). Overall the system has eleven states and two actions.

Safety Gym. We take the Static environment from (Yang et al., 2021), which is modifications of the safety gym environments (Ray et al., 2019). The unsafe region is placed near the goal and the robot is placed randomly, after the goal is reached the episode ends. The safety cost is incurred at every time step spent in the blue area. This environment is schematically depicted in Figure A1(d). We consider two robots: "point" with 46 states and 2 actions, and "car" with 56 states and 2 actions. Note that our variant of the safety gym environment has deterministic constraints, not randomly placed constraints as is common in other safety

Table A2. Default hyperparameters for SAC

Name	Value
Network architecture	[256,256]
Activation	ReLU
Value function learning rate	5e-4
Policy learning rate	5e-4
α learning rate	5e-2
Batch size	1024
Task Discount Factor	0.99
N samples per epochs	200
Training frequency	1
Target entropy	-$\lvert\mathcal{A}\rvert$
τ	0.005
Size of the replay buffer	1e6
N start updates	1e3
Penalty learning rate	5e-2
Safety Discount Factor	0.99

gym environments. We did so because our algorithm aims to deliver almost surely constraints and hence we did not want to contaminate our experiments with unsolvable problems.

E. Experiment details

We take the default parameters presented in Tables A1 and A2. For all modifications of TRPO/PPO and SAC, the default parameters and the code base is the same, which makes the direct comparisons fairer. Note that these parameters are used in the safety starter agents implementation of these algorithms (Ray et al., 2019).

Pendulum Swing-up. We use default parameters. We plot evaluation during training in Figures A2, A3, A4 and A5. Note that "sautéed" algorithms achieve a safe almost surely policy after 200 episodes of training. We plot maximum incurred cost over the episode to evaluate the constraint violation during training.

Double Pendulum We use default parameters for Vanilla TRPO, Sauté TRPO, and CPO. We run a hyper-parameter search for Lagrangian TRPO by varying the penalty learning rate (5e-3, 1e-2, 5e-2), backtrack iterations (10, 15, 20), value function learning rate (1e-4, 1e-3, 5e-3) and steps per epoch (4000, 10000, 20000). However, we did not find any parameter setting that performs significantly better than the default one. We plot evaluation during training in Figure A6. We plot the maximum incurred cost over the episode to evaluate the constraint violation during training. We also present results for ablation on the cost function in Table A3.

Reacher We use default parameters for Vanilla TRPO, Sauté TRPO, and CPO. We run a hyper-parameter search for Lagrangian TRPO by varying the penalty learning rate (5e-3, 1e-2, 5e-2), backtrack iterations (10, 15, 20), value function learning rate (1e-4, 1e-3, 5e-3) and steps per epoch (4000, 10000, 20000). We finally determined the following customized parameters after comparison: penalty learning rate (3e-2), backtracking iterations (15), value function learning rate (1e-2), steps per epoch (4000), and number of epochs(1000). Using this parameter setting makes the average cost fluctuates slightly around the safety budget, unlike the average cost of CPO which converges to the safety budget.

Table A3. Task (bold burgundy) and safety (italic blue) costs for Sauté TRPO and various reshaped costs $\widetilde{c}_n$. The first value in the bracket is 5% percent quantile, the second is the mean, and the the third is 95% percent quantile.

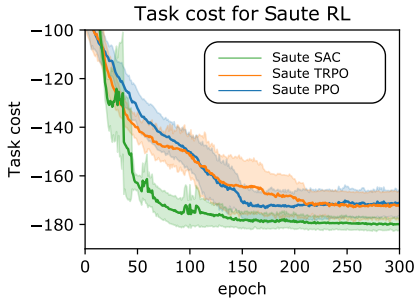
n	Task costs	Safety costs
0	$[\mathbf{-61.47}, \mathbf{-74.87}, \mathbf{-86.67}]$	$[32.49, 37.56, 39.68]$
10	$[\mathbf{-64.34}, \mathbf{-71.68}, \mathbf{-83.01}]$	$[36.84, 38.38, 39.47]$
100	$[\mathbf{-72.65}, \mathbf{-79.29}, \mathbf{-91.18}]$	$[35.91, 37.71, 39.69]$
1000	$[\mathbf{-80.24}, \mathbf{-83.95}, \mathbf{-89.16}]$	$[37.05, 38.15, 39.40]$
10000	$[\mathbf{-67.24}, \mathbf{-78.95}, \mathbf{-86.73}]$	$[36.94, 38.09, 39.50]$
100000	$[\mathbf{-67.19}, \mathbf{-76.42}, \mathbf{-85.81}]$	$[36.22, 37.92, 39.45]$

Safety Gym We use default parameters for Vanilla TRPO, Lagrangian TRPO, and CPO in both Point Goal and Car Goal. We run a hyper-parameter search for Sauté TRPO by varying the penalty learning rate (5e-3, 1e-2, 5e-2), backtrack iterations (10, 15, 20), value function learning rate (1e-4, 1e-3, 5e-3) and steps per epoch (4000, 6000, 10000, 15000). We finally determined the following customized parameters after comparison: penalty learning rate (3e-2), backtracking iterations (15), value function learning rate (5e-3), steps per epoch (10000), and the number of epochs(2000). Using this parameter setting keeps most of the costs below the safety budget while most of the task costs are similar to the task costs of other algorithms (see percentiles and medians in the box plots).

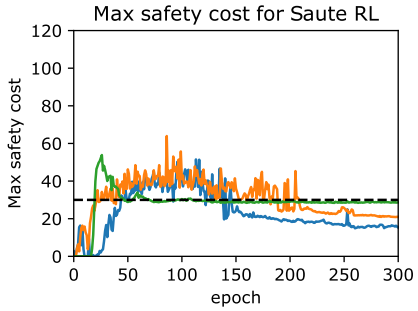
F. Further experiments

F.1. SAC with narrower networks

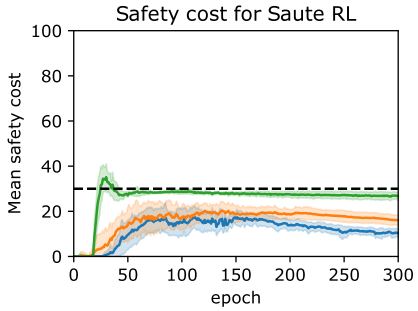
To compare directly PPO, TRPO, and SAC we performed another experiment while setting policy network architecture to $[64, 64]$ for SAC. We report the results in Figure A7. It is clear that Sauté SAC is still preferable to Lagrangian SAC in terms of safety almost surely, but increasing the width of the neural networks for its actors and critics improves the performance at the test time. In this work, we have not experimented more with SAC algorithms since TRPO and PPO-based are the basis for the state-of-the-art approaches.



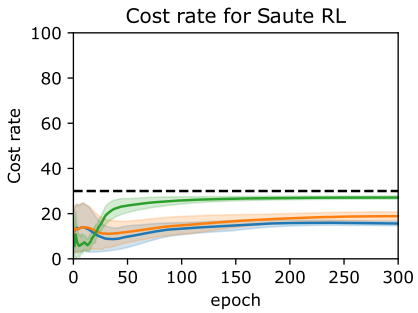
(a) Mean task costs



(b) Max safety costs

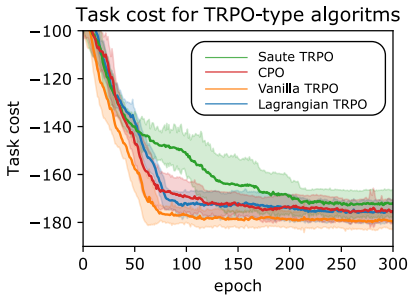


(c) Mean safety costs

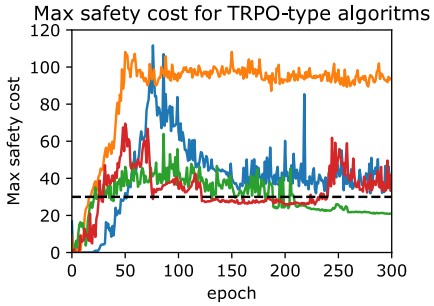


(d) Cost rates

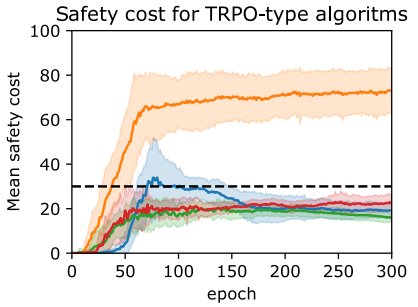
Figure A2. Evaluation results of Sauté PPO, Sauté TRPO and Sauté SAC on the pendulum swing-up task over 5 different seeds with 100 trajectories for every seed. Average task cost are depicted in Panel a (shaded areas are the standard deviation over all runs), maximum incurred safety costs are depicted in Panel b, average incurred safety costs are depicted in Panel c (shaded areas are the standard deviation over all runs), and cost rates are depicted in Panel d (shaded areas are the standard deviation over different seeds). The black-dotted line is the safety budget used for training.



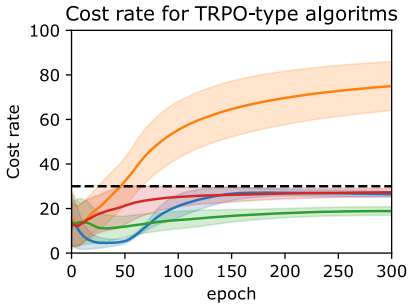
(a) Mean task costs



(b) Max safety costs

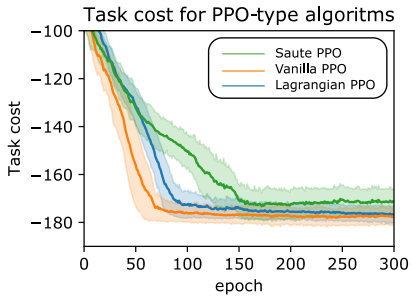


(c) Mean safety costs

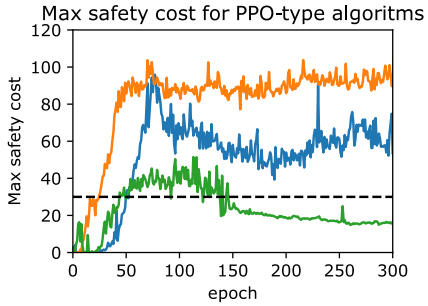


(d) Cost rate

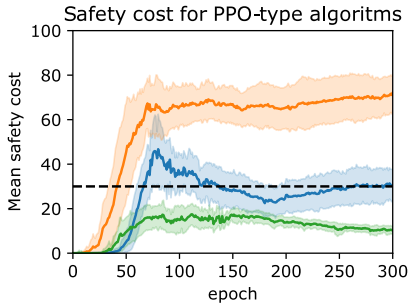
Figure A3. Evaluation results of Vanilla TRPO, Sauté TRPO, Lagrangian TRPO and CPO on the pendulum swing-up task over 5 different seeds with 100 trajectories for every seed. Average task cost are depicted in Panel a (shaded areas are the standard deviation over all runs), maximum incurred safety costs are depicted in Panel b, average incurred safety costs are depicted in Panel c (shaded areas are the standard deviation over all runs), and cost rates are depicted in Panel d (shaded areas are the standard deviation over different seeds). The black-dotted line is the safety budget used for training.



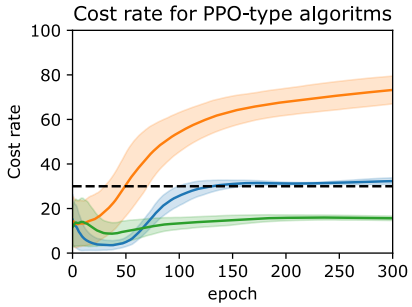
(a) Mean task costs



(b) Max safety costs

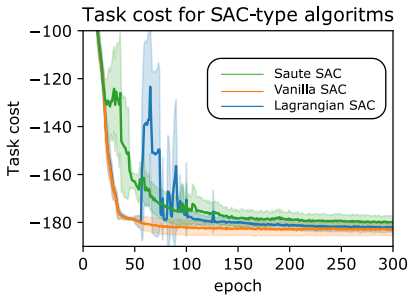


(c) Mean safety costs

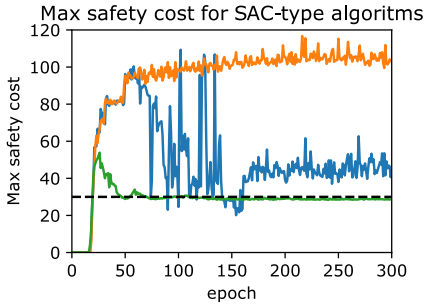


(d) Cost rate

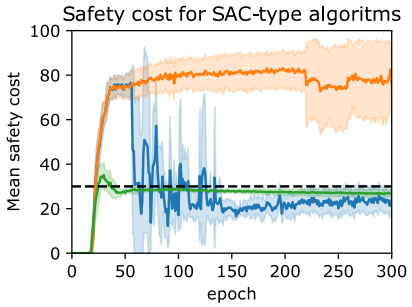
Figure A4. Evaluation results of Vanilla PPO, Sauté PPO and Lagrangian PPO on the pendulum swing-up task over 5 different seeds with 100 trajectories for every seed. Average task cost are depicted in Panel a (shaded areas are the standard deviation over all runs), maximum incurred safety costs are depicted in Panel b, average incurred safety costs are depicted in Panel c (shaded areas are the standard deviation over all runs), and cost rates are depicted in Panel d (shaded areas are the standard deviation over different seeds). The black-dotted line is the safety budget used for training.



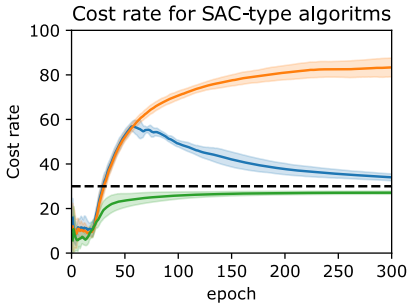
(a) Mean task costs



(b) Max safety costs

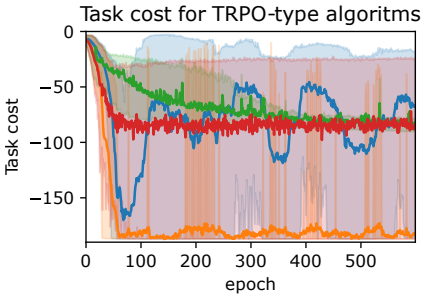


(c) Mean safety costs

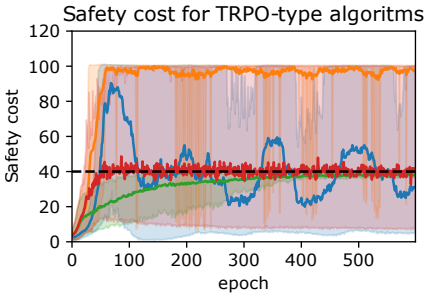


(d) Cost rate

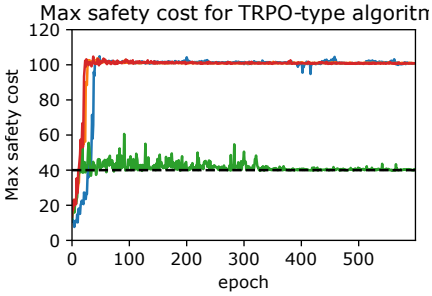
Figure A5. Evaluation results of Vanilla SAC, Sauté SAC and Lagrangian SAC on the pendulum swing-up task over 5 different seeds with 100 trajectories for every seed. Average task cost are depicted in Panel a (shaded areas are the standard deviation over all runs), maximum incurred safety costs are depicted in Panel b, average incurred safety costs are depicted in Panel c (shaded areas are the standard deviation over all runs), and cost rates are depicted in Panel d (shaded areas are the standard deviation over different seeds). The black-dotted line is the safety budget used for training.



(a) Mean task costs

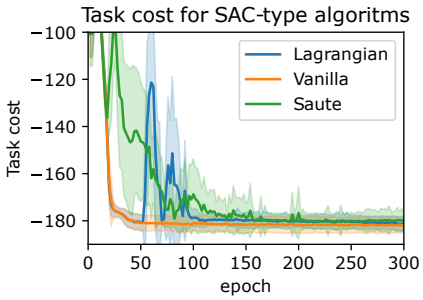


(b) Mean safety costs

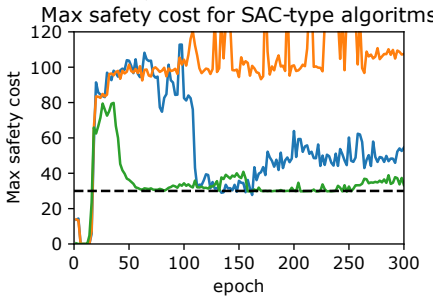


(c) Max safety costs

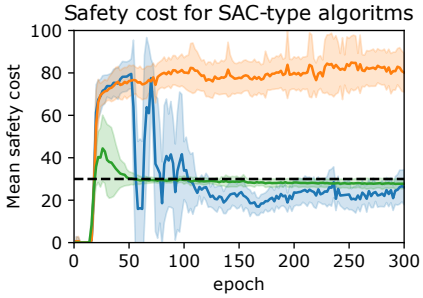
Figure A6. Evaluation results on the double pendulum environment. Vanilla TRPO, Sauté TRPO, Lagrangian TRPO and CPO are evaluated on 5 different seeds with 100 trajectories for every seed. Average task costs are depicted in Panel a (shaded areas are 80% percentile intervals), Average task costs are depicted in Panel b (shaded areas are 80% percentile intervals), maximum incurred safety costs are depicted in Panel c. The black-dotted line is the safety budget (40) used for training



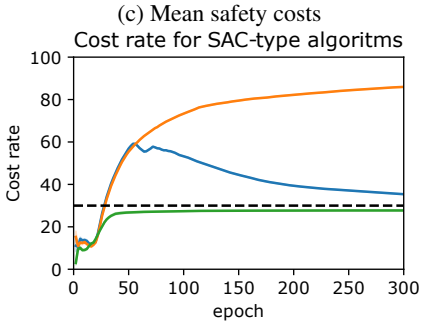
(a) Mean task costs



(b) Max safety costs



(c) Mean safety costs



(d) Cost rate

Figure A7. Evaluation results of Vanilla SAC, Sauté SAC and Lagrangian SAC on the pendulum swing-up task over 5 different seeds with 100 trajectories for every seed. Here we set the network architecture to $[64, 64]$. These results are comparable to the results in Figure A5, where the network architecture is $[256, 256]$

Table A4. Evaluation of Reacher during training. We report mean $\pm$ standard deviation for task costs; mean, 90% quantile, 99% quantile for safety costs. We havve used $K_i = 0.01$, $K_p = 0.1$ for PID-L, and $\alpha = 0.01$ for CVaR-TRPO.

	Task	Safety
Vanilla	-60.85 ± 0.10	$18.15, 19.14, 19.36$
Sauté	-61.35 ± 0.44	**4.51, 8.78, 9.12**
CPO	-60.88 ± 0.10	$9.95, 11.69, 13.18$
TRPO-L	-61.15 ± 0.44	$6.44, 12.44, 14.85$
CVaR-L	-60.91 ± 0.11	$7.81, 13.90, 17.41$
PID-L	-61.17 ± 0.13	$5.68, 10.02, 10.94$

Table A5. Evaluation of Point Goal during training. We report mean $\pm$ standard deviation for task costs; mean, 90% quantile, 99% quantile for safety costs. We havve used $K_i = 0.01$, $K_p = 0.1$ for PID-L, and $\alpha = 0.01$ for CVaR-TRPO.

	Task	Safety
Vanilla	2.89 ± 0.47	$16.18, 52.00, 55.08$
Sauté	2.77 ± 0.59	**0.19, 0.00, 0.00**
CPO	2.93 ± 0.49	$17.92, 52.00, 58.04$
TRPO-L	2.96 ± 0.45	$19.21, 52.00, 55.05$
CVaR-L	2.90 ± 0.48	$0.93, 0.00, 33.02$
PID-L	2.90 ± 0.60	$18.24, 52.00, 59.78$

F.2. Comparisons to PID-Lagrangian

Finally, we compare our method to the PID-Lagrangian approach (Stooke et al., 2020), and we do so on the same tasks and environments. However, instead of saving the policies and validating them separately, we use the training data as it was done in (Stooke et al., 2020). We do so to perform a fair comparison to (Stooke et al., 2020). We make the similar adjustments to the other baselines and present results in Tables A4, A5, A6. We did not observe a significant difference in the performance of PID-Lagrangian and TRPO-Lagrangian even after significant tuning efforts and hence our initial conclusions hold. Also, note that all Lagrangian algorithms did not converge on the Reacher environment.

Table A6. Evaluation of Car Goal during training. We report mean $\pm$ standard deviation for task costs; mean, 90% quantile, 99% quantile for safety costs. We havve used $K_i = 0.01$, $K_p = 0.1$ for PID-L, and $\alpha = 0.01$ for CVaR-TRPO.

	Task	Safety
Vanilla	2.96 ± 0.46	$19.88, 52.00, 68.02$
Sauté	2.81 ± 0.45	**0.22, 0.00, 0.00**
CPO	2.91 ± 0.46	$16.75, 53.00, 73.00$
TRPO-L	2.91 ± 0.47	$16.65, 51.00, 64.07$
CVaR-L	2.72 ± 0.64	$1.02, 0.00, 39.00$
PID-L	2.91 ± 0.61	$13.72, 48.00, 83.00$