SPoRt - Safe Policy Ratio: Certified Training and Deployment of Task Policies in Model-Free RL

Jacques Cloete[1], **Nikolaus Vertovec**[2] and **Alessandro Abate**[2]
[1]Oxford Robotics Institute, University of Oxford
[2]Department of Computer Science, University of Oxford
jacques@robots.ox.ac.uk, {nikolaus.vertovec, alessandro.abate}@cs.ox.ac.uk

Abstract

To apply reinforcement learning to safety-critical applications, we ought to provide safety guarantees during both policy training and deployment. In this work we present novel theoretical results that provide a bound on the probability of violating a safety property for a new task-specific policy in a model-free, episodic setup: the bound, based on a 'maximum policy ratio' that is computed with respect to a 'safe' base policy, can also be more generally applied to temporally-extended properties (beyond safety) and to robust control problems. We thus present SPoRt, which also provides a data-driven approach for obtaining such a bound for the base policy, based on scenario theory, and which includes Projected PPO, a new projection-based approach for training the task-specific policy while maintaining a user-specified bound on property violation. Hence, SPoRt enables the user to trade off safety guarantees in exchange for task-specific performance. Accordingly, we present experimental results demonstrating this trade-off, as well as a comparison of the theoretical bound to posterior bounds based on empirical violation rates.

1 Introduction

Reinforcement Learning (RL) is an area of machine learning where an agent is trained to interact with its environment to maximize some (cumulative) reward [Sutton and Barto, 2014; Mason and Grijalva, 2019]. There has been great interest in applying RL to real-world control problems in fields such as robotics [Kober and Peters, 2014; Hwangbo *et al.*, 2019; Singh *et al.*, 2022], traffic management [Chu *et al.*, 2020; Vertovec and Margellos, 2023; Lee *et al.*, 2023] and autonomous driving [Isele *et al.*, 2018; Ma *et al.*, 2021; Li *et al.*, 2022], to name just a few. Many of these domains typically fall into the realm of "safety-critical" applications, whereby we need to guarantee safety specifications, such as obstacle avoidance. Satisfying safety constraints becomes particularly challenging when we have little to no knowledge of our environment. This problem has been studied in a substantial body of literature, known as model-free safe RL.

Traditional policy gradient algorithms for model-free RL, such as Trust Region Policy Optimization (TRPO) [Schulman *et al.*, 2015] and Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017], allow the agent to explore any behavior during training, including behaviors that would be considered unsafe; this is unacceptable for safety-critical applications. To encode safety into training, a popular formulation is the Constrained Markov Decision Process (CMDP) [Altman, 2021], which includes safety constraints and is typically solved using primal-dual methods [Achiam *et al.*, 2017] and modifying the trust region to exclude unsafe policy updates [Milosevic *et al.*, 2024]. However, the CMDP formulation is limited in its ability to model safety constraints; CMDPs constrain the expected discounted cost return, but for many practical applications we require an explicit bound on the probability that a sampled trajectory violates a safety constraint.

Alternative approaches based on control theory ensure safety by preventing the agent from taking actions that would eventually lead to safety violations; this is achieved using, e.g., Lyapunov and barrier functions [Chow *et al.*, 2018], shielding [Alshiekh *et al.*, 2018; Konighofer *et al.*, 2023] or safety filters [Hsu *et al.*, 2024]. However, these approaches require a model of the environment to predict future safety, and thus are generally limited to model-based setups. Meanwhile, formal methods-based approaches, such as [Hasanbeig *et al.*, 2023], encode safety by leveraging Linear Temporal Logic (LTL) [Pnueli, 1977] as a formal reward-shaping structure. Unlike CMDPs, whereby the original objective is separate from the constraint, LTL formula satisfaction is encoded into the expected return itself, and under certain conditions the trained policy is guaranteed to maximize the probability of LTL formula satisfaction; however, no guarantees can be obtained *during* training.

Since we presume no knowledge of the environment, as in standard model-free RL, we will rely on finite-sample learning to evaluate the agent's ability to remain safe using probably approximately correct (PAC) guarantees. Finite-sample complexity bounds provide the number of samples needed to, with a given confidence, learn some target function with a certain accuracy [Vidyasagar, 2003]. Tools from statistical learning theory based on Vapnik Chervonenkis (VC) theory have successfully been able to provide finite sample bounds for learning in unknown environments [Vidyasagar, 2003; Tempo *et al.*, 2005], with recent work providing finite sample bounds even under changing target assumptions [Vertovec *et al.*, 2024]. Yet VC-theoretic techniques require the computation of the VC dimension, which is a difficult task for generic optimization problems. Under a convexity assumption, the so-called scenario approach offers a-priori probabilistic feasibility guarantees without resorting to VC theory [Calafiore and Campi, 2006; Campi and Garatti, 2008; Campi and Garatti, 2018].

The scenario approach traditionally relies on independent and identically distributed (i.i.d.) samples to establish its sample-complexity bounds. This creates a limitation in RL contexts, where the sampling distribution changes as policies are updated. As a result, safety guarantees established for one policy cannot be directly transferred when the policy changes. In this work, we overcome this limitation by extending the PAC guarantees to accommodate policy changes. Specifically, we derive a constraint on how much policies can shift while maintaining safety guarantees, and present SPoRt, an approach for adapting an existing safe policy to improve task-specific performance while maintaining a bound on the probability of safety violation, known prior to deploying or even training the adapted policy; this bound can be tuned by the user to trade off safety and task-specific performance.

Our technical contributions underpinning SPoRt are as follows:

1. A data-driven method for obtaining a bound on the probability that a property (e.g. safety), in general expressed as an LTL formula, is violated for trajectories drawn using a given 'safe' base policy (Section 3).
2. Novel theoretical results that provide, for an episodic, model-free RL setup, a prior bound on the probability of property violation for a new task-specific policy, based on a 'maximum policy ratio' computed with respect to the 'safe' base policy (cf. previous point) (Section 4).
3. A projection-based method for constraining the task-specific policy to ensure that this prior bound holds (Section 5).
4. Projected PPO, an algorithm for *training* a new, task-specific policy, while maintaining a user-specified prior bound on property violation, thus trading off safety guarantees for task-specific performance (Section 6).

We also test SPoRt on a time-bounded reach-avoid property and present experimental results demonstrating the safety-performance trade-off, as well as comparison of the theoretical prior bound to posterior bounds based on empirical violation rates (Section 7 and 8).

All appendices and code[1] can be found in the supplementary material, which contains all proofs.

2 Models, Tasks and Properties

We consider a model-free episodic RL setup where an agent interacts with an unknown environment modeled as a Markov Decision Process (MDP) [Sutton and Barto, 2014], specified by the tuple $\langle \mathcal{S}, \mathcal{A}, p, \mu, r_{\text{task}} \rangle$, with a continuous state space $\mathcal{S}$ and continuous action space $\mathcal{A}$. $p(s'|a,s) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ and $\mu(s) \in \Delta(\mathcal{S})$ are the (unknown) state-transition and initial state distributions, respectively. We will consider learning a stochastic policy $\pi(a|s) : \mathcal{S} \to \Delta(\mathcal{A})$ in a model-free setup. We use $\tau^{p,\pi}_{\mathbf{s}_t,T} = (\mathbf{s}_t, \mathbf{s}_{t+1}, \ldots, \mathbf{s}_{t+T})^{p,\pi}$ to denote a realization of a trajectory of the closed-loop system with state transition distribution p, starting at state $\mathbf{s}_t$ and evolving for T time steps, using policy π. $r_{\text{task}}(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the task-specific reward, which encourages higher task-specific performance (for example, max speed or min time).

2.1 Safety as a Temporally-Extended Property

We define safety in terms of satisfaction of a general temporal property φ. We denote that a trajectory τ satisfies property φ (and is therefore safe) by $\tau \models \varphi$, while $\tau \not\models \varphi$ indicates that τ violates φ (and is therefore unsafe). SPoRt addresses problems where the objective is to ensure that $\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T} \models \varphi$ with high probability, while maximizing the task-specific reward r_{task}. To evaluate the satisfaction of φ we introduce a robustness metric ϱ^{φ}, which encodes property violation as a real-valued signal that is non-negative only when $\tau \models \varphi$.

Definition 1. A *robustness metric* ϱ^{φ} is a function $\varrho^{\varphi}(\tau) : \mathcal{S}^n \to [-a, b],\ n \in \mathbb{Z}_+,\ a, b \in \mathbb{R}_+$ such that $\varrho^{\varphi}(\tau) \geq 0$ only for trajectories $\tau \in \mathcal{S}^n$ that satisfy property φ (i.e. $\tau \models \varphi$).

Any safety property φ can be expressed as a Linear Temporal Logic (LTL) formula [Pnueli, 1977], which ensures the existence of such a metric (see Appendix A.1). Notably, SPoRt extends beyond safety properties to encompass any property φ expressible as an LTL formula - the case study deals with 'reach-avoid' as we shall see. Accordingly, our theoretical results generalize to product MDPs in RL problems under general LTL specifications [Hasanbeig *et al.*, 2023]. While Appendix A.2 provides detailed discussions on these extensions to general LTL formulae and hybrid-state models, for the remainder of the paper (and with no loss in generality) we focus exclusively on safety properties φ within continuous-state MDPs, as defined in Section 2.

3 Data-Driven Property Satisfaction

SPoRt provides a method for adapting an existing safe policy (π_{base}) so as to maximize some task-specific reward (r_{task}), without violating a given property φ.

As a first step, let us evaluate the property satisfaction of given traces for a general policy π. Given an initial state distribution, state transition distribution and stochastic policy (μ, p, π), the value of the robustness metric for an associated trajectory, i.e., $\varrho^{\varphi}(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T})$ will be a random variable drawn from some distribution $\Delta^{p,\pi}_{\mu}$ and the probability of satisfying the property φ will be encoded by

$$\mathbb{P}\{\varrho^{\varphi}(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T}) \in \Delta^{p,\pi}_{\mu} : \varrho^{\varphi}(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T}) \geq 0\}.$$

SPoRt first bounds the probability of property violation under an existing 'safe' base policy π_{base}, i.e, $\mathbb{P}\{\varrho^{\varphi}(\tau^{p,\pi_{\text{base}}}_{\mathbf{s}_0\sim\mu,T}) \in \Delta^{p,\pi_{\text{base}}}_{\mu} : \varrho^{\varphi}(\tau^{p,\pi_{\text{base}}}_{\mathbf{s}_0\sim\mu,T}) < 0\} \leq \epsilon_{\text{base}}$ using the scenario approach [Campi and Garatti, 2018]; we roll out N scenario trajectories $(\tau^{p,\pi_{\text{base}}}_{\mathbf{s}_0\sim\mu,T})_i$ using π_{base} and record them in buffer $\mathcal{D}^{p,\pi_{\text{base}}}_{\mu} = \{(\tau^{p,\pi_{\text{base}}}_{\mathbf{s}_0\sim\mu,T})_i\}_{i=1}^{N}$. We use Theorem 1 to obtain an upper bound ϵ_{base} on the probability of violating φ:

[1] Link to code: https://github.com/JacquesCloete/sport.

Theorem 1. *If $\varrho^\varphi((\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T})_i) \geq 0$ for all N scenarios $(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T})_i$ in $\mathcal{D}^{p,\pi}_\mu = \{(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T})_i\}_{i=1}^N$, then with confidence $1-\beta$, where $\beta = (1-\epsilon)^N$, the probability of drawing a new scenario $\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T}$ such that $\varrho^\varphi(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T}) < 0$ is at most ϵ.*

If not all scenarios in $\mathcal{D}^{p,\pi_{\text{base}}}_\mu$ satisfy φ, we can leverage results from [Campi and Garatti, 2010] to identify a suitable ϵ_{base} 'under k-constraint removal', as follows:

Corollary 1. *Assume k scenarios $(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T})_i$ in buffer $\mathcal{D}^{p,\pi}_\mu$ = $\{(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T})_i\}_{i=1}^N$ are such that $\varrho^\varphi((\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T})_i) < 0$, then with confidence $1-\beta$, where $\beta = \sum_{i=0}^k \binom{N}{i}\epsilon_k^i(1-\epsilon_k)^{N-i}$, the probability of drawing a new scenario $\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T}$ such that $\varrho^\varphi(\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T}) < 0$ is at most ϵ_k.*

In both cases, we first collect N scenarios, then choose our confidence $1-\beta$, and then compute the bound ϵ_{base}.

4 Property Violation under Modified MDPs

Once ϵ_{base} is obtained, SPoRt safely trains a task-specific policy π_{task} so as to maximize the (cumulative) reward r_{task}. For SPoRt to ensure safe training of π_{task}, we must upper bound the probability of property violation under π_{task}, i.e., $\mathbb{P}\{\varrho^\varphi(\tau^{p,\pi_{\text{task}}}_{\mathbf{s}_0\sim\mu,T}) \in \Delta^{p,\pi_{\text{task}}}_\mu : \varrho^\varphi(\tau^{p,\pi_{\text{task}}}_{\mathbf{s}_0\sim\mu,T}) < 0\}$, by the probability of property violation under π_{base}, i.e., $\mathbb{P}\{\varrho^\varphi(\tau^{p,\pi_{\text{base}}}_{\mathbf{s}_0\sim\mu,T}) \in \Delta^{p,\pi_{\text{base}}}_\mu : \varrho^\varphi(\tau^{p,\pi_{\text{base}}}_{\mathbf{s}_0\sim\mu,T}) < 0\}$, which is upper bounded by ϵ_{base}. To do so, we first construct this bound for general (μ_1, p_1, π_1) and (μ_2, p_2, π_2), and then set $(\mu_1, p_1, \pi_1) = (\mu, p, \pi_{\text{base}})$ and $(\mu_2, p_2, \pi_2) = (\mu, p, \pi_{\text{task}})$.

Let $\mathcal{S}_0, \ldots, \mathcal{S}_T \subseteq \mathcal{S}$ be a sequence of arbitrary subsets of the state space for each time step in the episode. We begin by characterizing the probability that a sampled trajectory $\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T} = (\mathbf{s}_0 \sim \mu, \mathbf{s}_1, \ldots, \mathbf{s}_T)^{p,\pi}$ is such that $\mathbf{s}_0 \in \mathcal{S}_0, \ldots, \mathbf{s}_T \in \mathcal{S}_T$ as a forward recursion, based on work in [Soudjani and Abate, 2013; Soudjani and Abate, 2015]. Let $\mathbb{1}_{\mathcal{S}_t}(s) : \mathcal{S} \to \{0,1\}$ be the indicator function for $s \in \mathcal{S}_t$, and define functions $W_t^{\mu,p,\pi}(s) : \mathcal{S} \to \mathbb{R}_+$, characterized as

$$W_{t+1}^{\mu,p,\pi}(s') = \mathbb{1}_{\mathcal{S}_{t+1}}(s') \int_{\mathcal{S}} P^{p,\pi}(s'|s) W_t^{\mu,p,\pi}(s) ds$$
$$\text{and} \quad W_0^{\mu,p,\pi}(s') = \mathbb{1}_{\mathcal{S}_0}(s')\mu(s'),$$
$$\text{where} \quad P^{p,\pi}(s'|s) = \int_{\mathcal{A}} p(s'|a,s)\pi(a|s) da.$$

It holds that $\mathbb{P}\{\tau^{p,\pi}_{\mathbf{s}_0\sim\mu,T} : \mathbf{s}_0 \in \mathcal{S}_0, \ldots, \mathbf{s}_T \in \mathcal{S}_T\} = \int_{\mathcal{S}} W_T^{\mu,p,\pi}(s) ds$. We then use Theorem 2 to bound the probability that trajectory $\tau^{p_2,\pi_2}_{\mathbf{s}_0\sim\mu_2,T}$ remains within $\mathcal{S}_0, \ldots, \mathcal{S}_T$ throughout the episode in terms of the probability that trajectory $\tau^{p_1,\pi_1}_{\mathbf{s}_0\sim\mu_1,T}$ remains within the same $\mathcal{S}_0, \ldots, \mathcal{S}_T$:

Theorem 2. *Suppose that, for a set of coefficients $\alpha_t \in \mathbb{R}_+$, we could constrain μ_2, p_2 and π_2 so as to enforce the following bounds for all $t = 1, \ldots, T$:*

$$\begin{aligned} &\int_{\mathcal{S}} P^{p_2,\pi_2}(s'|s) W_{t-1}^{\mu_1,p_1,\pi_1}(s) ds \\ \leq &\alpha_t \int_{\mathcal{S}} P^{p_1,\pi_1}(s'|s) W_{t-1}^{\mu_1,p_1,\pi_1}(s) ds \qquad (1) \\ \textit{and} \quad &\mu_2(s') \leq \alpha_0 \mu_1(s'), \quad \forall s' \in \mathcal{S}. \end{aligned}$$

It thus holds that

$$\begin{aligned} &\mathbb{P}\{\tau^{p_2,\pi_2}_{\mathbf{s}_0\sim\mu_2,T} : \mathbf{s}_0 \in \mathcal{S}_0, \ldots, \mathbf{s}_T \in \mathcal{S}_T\} \\ \leq &\mathbb{P}\{\tau^{p_1,\pi_1}_{\mathbf{s}_0\sim\mu_1,T} : \mathbf{s}_0 \in \mathcal{S}_0, \ldots, \mathbf{s}_T \in \mathcal{S}_T\} \prod_{t=0}^T \alpha_t. \end{aligned}$$

We want to make this bound as tight as possible, which is done by minimizing α_t subject to Equation (1) for all $s' \in \mathcal{S}$ and $t = 1, \ldots, T$. Solving this problem is non-trivial due to p_1 and p_2 being unknown in a model-free setup. To this end, we introduce Theorem 3 to obtain a feasible solution by constraining p_2 and π_2 in terms of p_1 and π_1:

Theorem 3. *Suppose the following constraint holds:*

$$p_2(s'|a,s)\pi_2(a|s) \leq \alpha_t p_1(s'|a,s)\pi_1(a|s) \; \forall a \in \mathcal{A}, s \in \mathcal{S}.$$

Thus Equation (1) *holds for all $s' \in \mathcal{S}$.*

Assuming stationarity, under this constraint the bound is minimized when $\alpha_t = \alpha$ for all $t = 1, \ldots, T$. Note also that under this constraint, we find that $\alpha \geq 1$.

It is important to observe that, while correct, this bound can be very conservative for applications with large episode length T; a discussion on this conservativeness can be found in Appendix C.1. Alternative bounds from literature suffer from similar blowup [Soudjani and Abate, 2012; Soudjani and Abate, 2015].

Using the results from Theorem 2 and 3, we can now derive Theorem 4 to obtain a bound on the probability of property violation for (μ_2, p_2, π_2) in terms of (μ_1, p_1, π_1):

Theorem 4. *Suppose that*

$$\mathbb{P}\{\varrho^\varphi(\tau^{p_1,\pi_1}_{\mathbf{s}_0\sim\mu_1,T}) \in \Delta^{p_1,\pi_1}_{\mu_1} : \varrho^\varphi(\tau^{p_1,\pi_1}_{\mathbf{s}_0\sim\mu_1,T}) < 0\} \leq \epsilon_1$$

and for all $t = 1, \ldots, T$,

$$\begin{aligned} &p_2(s'|a,s)\pi_2(a|s) \leq \alpha_t p_1(s'|a,s)\pi_1(a|s) \\ \textit{and} \quad &\mu_2(s') \leq \alpha_0 \mu_1(s'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S}. \end{aligned}$$

It thus holds that

$$\mathbb{P}\{\varrho^\varphi(\tau^{p_2,\pi_2}_{\mathbf{s}_0\sim\mu_2,T}) \in \Delta^{p_2,\pi_2}_{\mu_2} : \varrho^\varphi(\tau^{p_2,\pi_2}_{\mathbf{s}_0\sim\mu_2,T}) < 0\} \leq \epsilon_1 \prod_{t=0}^T \alpha_t.$$

The proof for Theorem 4 considers all sequences $\mathcal{S}_0, \ldots, \mathcal{S}_T$ that correspond to property violation events, and sums over their probabilities under (μ_2, p_2, π_2).

Now let $(\mu_1, p_1, \pi_1) = (\mu, p, \pi_{\text{base}})$ and $(\mu_2, p_2, \pi_2) = (\mu, p, \pi_{\text{task}})$; we see that constraining the policy ratio $\frac{\pi_{\text{task}}(a|s)}{\pi_{\text{base}}(a|s)} \leq \alpha$ for all $a \in \mathcal{A}, s \in \mathcal{S}$ is sufficient to ensure that the bound holds. The total multiplicative increase on the upper bound for property violation going from π_{base} to π_{task} is thus α^T, with the bound being $\epsilon_{\text{task}} = \epsilon_{\text{base}}\alpha^T$.

This is a significant result, since we can now provide a prior bound on the probability of property violation for *any* π_{task}, based *entirely* on the probability of property violation for π_{base} and the maximum policy ratio between π_{task} and π_{base} across all states and actions, with no required knowledge of μ, p or the constraints under which property φ holds, so long as initial state distribution and state transition distribution remain the same. Furthermore, by adjusting the value of α we can directly trade off safety guarantees for deviation from the base policy, which can be leveraged to achieve a boost in task-specific performance.

However, note the exponential relationship between T and ϵ_{task}; this means that, for even small increases of α from 1, our prior bound will always eventually explode to the point of becoming trivially 1 if T is made sufficiently large. Thus, if the prior bound is to be used, SPoRt is best suited to control problems with a low maximum episode length T. In practice, however, there are ways to overcome or otherwise mitigate this limitation, as we will see later in Section 7.

Note that Theorem 4 also provides a bound when μ_2 and p_2 differ from μ_1 and p_1. Thus, our theoretical results can also be applied to *robust control* settings for perturbed systems; see Appendix C.2 for further discussion.

5 Constraint Satisfaction for a Task Policy

For Theorem 4 to hold, we must maintain the hard constraint $\frac{\pi_{\text{task}}(a|s)}{\pi_{\text{base}}(a|s)} \leq \alpha$ for all $a \in \mathcal{A}, s \in \mathcal{S}$. Given a π_{task}, we can achieve this by projecting π_{task} onto the feasible set of policy distributions $\mathit{\Pi}_{\alpha,\pi_{\text{base}}}$ at each time step:

$$\pi_{\text{proj}}(a|s) = \text{proj}_{\mathit{\Pi}_{\alpha,\pi_{\text{base}}}(s)}(\pi_{\text{task}}(a|s)),$$
$$\text{where } \mathit{\Pi}_{\alpha,\pi_{\text{base}}}(s) = \left\{\pi : \alpha \geq \frac{\pi(a|s)}{\pi_{\text{base}}(a|s)} \;\forall a \in \mathcal{A}\right\}.$$

While π_{task} represents the unconstrained (and potentially unsafe) task-specific policy network that we train or are provided, the projection π_{proj} is a policy that we can safely roll out, including during training. Note that α defines the level sets of $\mathit{\Pi}_{\alpha,\pi_{\text{base}}}$, which is always non-empty for $\alpha \geq 1$ (since π_{base} itself is a valid π_{task}). Let us now look at how to simplify the computation of this projection step – we will henceforth assume diagonal Gaussian policies:

Assumption 1. Both π_{base} and π_{task} are diagonal Gaussian policies: $\pi(\mathbf{a}|s) = \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \text{diag}\left(\boldsymbol{\sigma}^2\right)$, and where the policy means $\boldsymbol{\mu}(s)$ and standard deviations $\boldsymbol{\sigma}(s)$ are functions of MDP state and evaluated at each time step using (for example) a policy neural network.

At each time step, we can obtain π_{proj} using Theorem 5:

Theorem 5. *Assuming diagonal Gaussian policies and using KL divergence as the distance metric for projection, the means and standard deviations of projected policy π_{proj} can be computed from π_{base} and π_{task} by solving the following convex optimization problem at each time step:*

$$\begin{aligned}
\min_{\boldsymbol{\mu}_{\text{proj}},\boldsymbol{\sigma}_{\text{proj}}} \quad & J(\boldsymbol{\mu}_{\text{proj}}, \boldsymbol{\sigma}_{\text{proj}}) \\
\text{subject to} \quad & \prod_{i=1}^{n}\left(\frac{\sigma_{\text{base},i}}{\sigma_{\text{proj},i}} e^{\frac{1}{2}\frac{\left(\mu_{\text{proj},i}-\mu_{\text{base},i}\right)^2}{\sigma^2_{\text{base},i}-\sigma^2_{\text{proj},i}}}\right) \leq \alpha, \\
& 0 < \sigma_{\text{proj},i} < \sigma_{\text{base},i} \quad \forall i = 1, \ldots, n
\end{aligned}$$

where $J(\boldsymbol{\mu}_{\text{proj}}, \boldsymbol{\sigma}_{\text{proj}})$

$$= \sum_{i=1}^{n}\left(-2\ln\left(\sigma_{\text{proj},i}\right) + \frac{\sigma^2_{\text{proj},i}}{\sigma^2_{\text{task},i}} + \frac{\left(\mu_{\text{proj},i} - \mu_{\text{task},i}\right)^2}{\sigma^2_{\text{task},i}}\right).$$

It is interesting to note that the standard deviations of π_{proj} must all be strictly lower than those of π_{base}, in other words we require π_{proj} to be less exploratory than π_{base}. This is intuitive considering that we aim to maintain safety by remaining 'close' to π_{base}. We also note that making π_{base} more exploratory (with a larger standard deviation) generally results in a larger $\mathit{\Pi}_{\alpha,\pi_{\text{base}}}$, allowing for greater policy change and thus task-specific performance boost by π_{proj}; see Appendix C.3 for details.

We implement and solve this problem using CVXPY [Diamond and Boyd, 2016; Agrawal *et al.*, 2018]; on a standard desktop PC the compute time remains in the order of milliseconds for even high-dimensional action spaces, making this method feasible for many RL applications. See Appendix D for implementation details.

6 Training for Tasks, Under a Bound on Property Violation

Above, we have shown how to obtain π_{proj} from π_{base} and π_{task}. We have implicitly assumed that we already have π_{base} and a corresponding bound on probability of property violation ϵ_{base}. There are many practical applications where we would also have access to π_{task}: as an example from robotics, we may have trained π_{task} in simulation, where safety is non-critical, but now want to safely deploy this policy on the real robot, for which a tried-and-tested π_{base} is known.

However, other applications may require that we train π_{task} to maximize cumulative r_{task} while maintaining a prior bound on safety during training, for example if a suitable simulator to train good policies for the real environment is not available. In this case, we would first choose an acceptable $\alpha \geq 1$ by rearranging $\epsilon_{\text{task}} = \epsilon_{\text{base}}\alpha^T \leq \epsilon_{\max}$, where $\epsilon_{\max}$ is a maximum acceptable probability of property violation, and we then learn π_{task} (initialized as π_{base}) while only ever deploying π_{proj} during training so as to ensure the bound holds.

Note that there is a distinction between what we train, π_{task}, and what we actually deploy, π_{proj}, during training. To overcome this, SPoRt uses Projected PPO, outlined in Algorithm 1, to train π_{task} for tasks with continuous state-action spaces. The algorithm is inspired by clipped PPO [Schulman *et al.*, 2017] but clips the surrogate advantage based on the policy ratio between the new π_{task} and previous π_{proj}, rather than the previous π_{task}; we store the (log-)probabilities of π_{proj} at each time step during data collection to avoid needing to recompute π_{proj} during gradient updates. The advantage estimates are also computed using samples collected by deploying π_{proj} rather than π_{task}.

The clipping sets the gradient to zero beyond the maximum/minimum allowed policy ratio of π_{task} to π_{proj}, preventing π_{task} from drifting away beyond clipping ratio ξ of π_{proj}, in this way, we maintain an acceptable amount of mismatch between π_{task} and π_{proj} and stop π_{task} from drifting away from the feasible set of allowed policy distributions.

We also warm-start the value function network for r_{task} before training π_{task}, since an accurate value function is important for effective fine-tuning. This is done by training the value function using clipped PPO until convergence while keeping the policy network weights fixed as those of π_{base}.

Algorithm 1 Projected PPO

Input: $\theta_{\text{base}}, \alpha, T$

Obtain initial critic parameters ϕ_0 by warm-starting the critic using PPO (episode length T) with π_{base}

Initial task-specific policy parameters $\theta_0 \leftarrow \theta_{\text{base}}$

for $k = 0, 1, 2, \ldots$ **do**

Collect trajectories $\mathcal{D}_k = \{(\tau_T)_i\}$ by running projected policy $\pi_{\text{proj},\theta_k} = \text{proj}_{\Pi_{\alpha,\pi_{\text{base}}}}(\pi_{\text{task},\theta_k})$ in the environment. Store $\pi_{\text{proj},\theta_k}(\cdot|\mathbf{s}_t)\ \forall \mathbf{s}_t \in \tau_T, \tau_T \in \mathcal{D}_k$

Compute rewards-to-go $\hat{R}_t$

Compute advantage estimates $\hat{A}_t$ using V_{ϕ_k}

Update task-specific policy:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\Big\{ \frac{\pi_{\text{task},\theta}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\text{proj},\theta_k}(\mathbf{a}_t|\mathbf{s}_t)} A^{\pi_{\text{proj},\theta_k}}(\mathbf{s}_t, \mathbf{a}_t),\ g\big(\xi, A^{\pi_{\text{proj},\theta_k}}(\mathbf{s}_t, \mathbf{a}_t)\big) \Big\}$$

$$\text{where } g(\xi, A) = \begin{cases} (1+\xi)A & \text{if } A \geq 0 \\ (1-\xi)A & \text{if } A < 0 \end{cases}$$

Fit value function:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \Big(V_\phi(\mathbf{s}_t) - \hat{R}_t\Big)^2$$

end for

7 SPoRt in Action: Case Studies

We apply SPoRt to a *reach-avoid* property, wherein the agent must reach a goal within a time limit while avoiding collision with a hazard up until the goal is reached. Such an objective is standard within the control and verification literature, and indeed can be used to model many real-world problems. For completeness, we provide the LTL formula and robustness metric for the time-bounded reach-avoid property in Appendix E.1, with a reminder that SPoRt can be used over general LTL specifications.

We implemented the environment in Safety Gymnasium [Ji *et al.*, 2023] using a point agent and with the goal and hazard being green and red circular regions, respectively (cf. Figure 1 and 2). This setup was chosen since it allows for easy interpretability of results while remaining a reasonable abstraction of a real robotic navigation task using a skid-steering mobile robot with a LiDAR sensor; a description of the MDP observation and action spaces can be found in Appendix E.2. The episode is reset if the agent enters the hazard or goal sets, or if the maximum episode length is exceeded. To mitigate the exponential relationship between maximum episode length T and bound ϵ_{task}, we reduced the control frequency ten-fold from the default (up to 100 simulation steps per environment step), in-keeping with the observation in Section 4 that SPoRt is best suited to control problems with low T.

π_{base} was trained so as to achieve a high probability of satisfying the property (reach-avoid), while remaining fairly exploratory. This was achieved by training π_{base} using Soft Actor-Critic (SAC) [Haarnoja *et al.*, 2018] with a *sparse* reward scheme (corresponding to property satisfaction across an *entire* episode). Alternative synthesis schemes are possible. Further details on training π_{base} can be found in Appendix E.3. Once trained, around $N = 10000$ scenarios were collected to determine ϵ_{base} with high confidence ($\beta = 1\text{e}{-7}$, see [Campi and Garatti, 2018]) using Corollary 1. Note that while training π_{base} the maximum episode length was set to $T = 100$ yet by the end of training the average episode length was much lower, at around $T = 14$. Thus, to keep the value of $\epsilon_{\text{task}} = \epsilon_{\text{base}}\alpha^T$ as low as possible, the maximum episode length was reduced to $T = 21$ after training π_{base} (with ϵ_{base} computed using scenarios of this length). Further discussions (including how SPoRt can be modified to do this automatically) can be found in Appendix E.4.

For our case studies we trained π_{task} to reach the goal as quickly as possible: accordingly, r_{task} was the standard dense reward for reaching a goal used by Safety Gymnasium. Notice that the set task (and corresponding reward) clearly leads to a potential violation of the property (reach-avoid) of interest. We consider two separate cases, as follows:

Case 1: Pre-Trained Task Policy. π_{task} is trained separately without any consideration for property violation. As a result, under π_{task} the agent quickly drives directly towards the goal with no hazard avoidance. This represents applications where π_{task} has been pre-trained in an environment where safety is not critical (for example in a robotics simulator) and we want to safely test it on the real environment (see Section 5).

Case 2: Task Policy Trained Using Projected PPO. This represents applications where we have π_{base} and ϵ_{base} (as from above) and now want to fine-tune our policy to be faster (thus obtaining π_{task}), while maintaining an acceptable given bound on property violation (see Section 6).

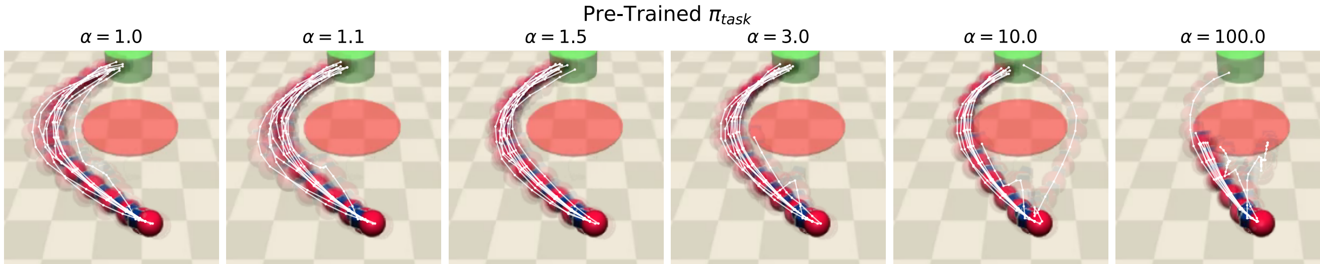
Note that we use the same π_{base} for both cases.

8 SPoRt Report: Results and Discussion

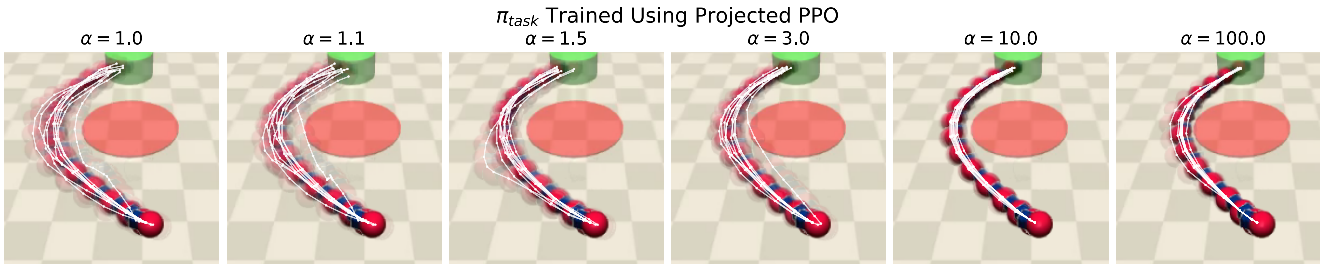
Both cases were tested for 1000 episodes at different values of α (such that $\pi_{\text{proj}} = \text{proj}_{\Pi_{\alpha,\pi_{\text{base}}}}(\pi_{\text{task}})$), ranging from $\alpha = 1$ (i.e. $\pi_{\text{proj}} = \pi_{\text{base}}$) to the point where the empirical violation rate exceeded a threshold. For Case 2, π_{task} was trained until convergence at each value of α, prior to testing. Further details on training π_{task} for both cases can be found in Appendix E.3. Action seeding for each episode was controlled across different values of α and across the different cases, so all results depend on α and the training of π_{task}. From our results we seek to answer the following questions (Qs):

1. Does increasing α trade off safety for performance?
2. How does performance compare between Case 1 and 2?
3. How conservative is the prior bound $\epsilon_{\text{task}} = \epsilon_{\text{base}}\alpha^T$?

Figure 1 presents sample distributions of episode trajectories for both cases over different values of α. In both cases, we see that as α increases, the trajectories bend more tightly



(a)



(b)

Figure 1: Sample distributions of episode trajectories from the reach-avoid experiment using π_{proj} for different values of α. (1a) Case 1 (pre-trained π_{task}). (1b) Case 2 (π_{task} trained using Projected PPO). Action seeding for each episode was controlled across different values of α and across the different cases, so all results depend on α and the training of π_{task}.

around the hazard, suggesting a reduction in action variance,[2] as well as an action mean that takes the agent closer to the hazard. In fact, in Case 1 for $\alpha = 100$, the agent's mean trajectory crosses the hazard. Thus increasing α is shown to trade off safety for performance, answering Q1.

However, we can also appreciate the difference between Case 1 and Case 2: while Case 1 produces an action mean that drives the agent through the hazard for $\alpha = 100$, Case 2 instead produces a more reduced action variance, while retaining an action mean that keeps the agent outside the hazard, as expected. Thus, to answer Q2, we see that Case 2 allows us to provide better performance, whilst retaining a 'better behaved' (and indeed 'safe') π_{proj} compared to Case 1; accordingly, we argue that since training π_{task} using Projected PPO deploys π_{proj} during training, π_{task} learns to optimize performance of π_{proj} compared to naïvely training π_{task} a priori with no consideration of how π_{proj} will perform.

Figure 2 presents a more detailed view of the agent behavior over an example episode for Case 2, for $\alpha = 5$ (representing a compromise between safety and performance). Looking at mean turning velocity over the episode, we see that while both π_{base} and π_{task} drive the agent clockwise around the hazard, π_{task} induces sharper turning, taking the agent closer to the hazard and drawing a tighter, shorter curve while maintaining the same or faster forward drive force. However, this sharper turning is constrained such that π_{proj} always lies within the $\alpha = 5$ level set (see Section 5). Note at π_{task} applies a reduced forward drive force at the very start of the episode compared to π_{base}, which makes sense given the agent is initially pointing away from the goal, so π_{task} reduces episode length by first pointing the agent closer to the agent before driving forward. Appendix E.5 provides a similar analysis for Case 1.

Figure 3a presents violation probabilities over different values of α for both cases. The most striking observation is the conservativeness of prior bound ϵ_{task}, which grows exponentially from $\epsilon_{\text{base}} = 0.009$ at $\alpha = 1$ to $\epsilon_{\text{task}} = 1$ at around $\alpha = 1.25$ (beyond which point the bound is no longer useful), yet the posterior bounds on property violation (obtained by applying Corollary 1 to the $N = 1000$ test samples) remain at around 0.025 over this range. We do also see exponential growth in the posterior bound for Case 1, but this happens over a completely different scale ($\alpha = 1$ to 100 rather than 1 to 1.25). The posterior bound for Case 2 remains at 0.025 for even $\alpha = 100$, suggesting much safer behavior compared to Case 1 for the same α.

Figure 3b presents the mean and standard deviation episode length for successful trajectories over different values of α for both cases. For both we see a similar reduction in mean and standard deviation as α increases until around $\alpha = 10$, at which point the mean plateaus (to around 12.0 at $\alpha = 100$, for 14.3% total reduction) but standard deviation shrinks for Case 2 while the mean continues to decrease for Case 1 (to around 11.3 at $\alpha = 100$, for 19.3% total reduction)); this comes at the cost of substantially increased violation rate for Case 1, shown by Figure 3a. Another important observation is that while we know from Figure 3a that ϵ_{task} is very conservative, we do see a measurable (2.1%) reduction in mean episode length for both cases from 14.0 at $\alpha = 1$ ($\epsilon_{\text{task}} = 0.009$) to around 13.7 at $\alpha = 1.12$ ($\epsilon_{\text{task}} = 0.1$, a fairly sensible (if high) value). Appendix E.5 provides the same figures but zoomed in to the scale across which $\epsilon_{\text{task}} \leq 1$.

[2] Recall we work with diagonal Gaussian policies, hence the consideration of action mean and variance.

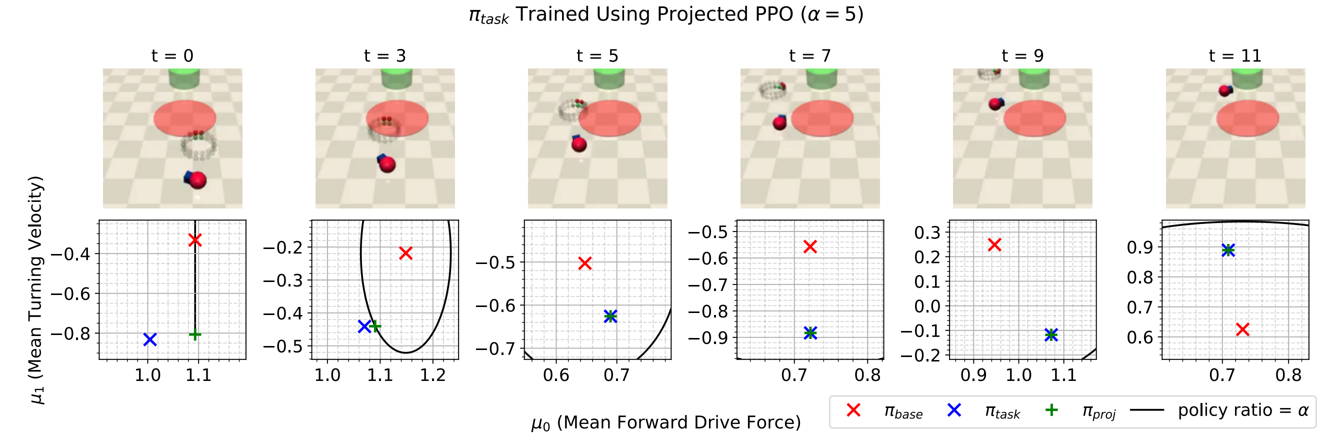
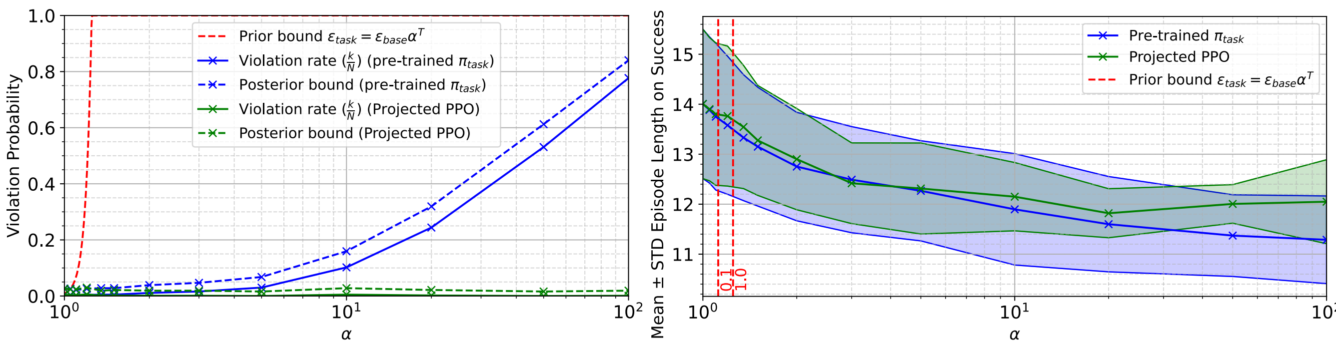


Figure 2: Snapshots across an example episode of the reach-avoid experiment using π_{proj} for Case 2 (π_{task} trained using Projected PPO) and $\alpha = 5$; the bottom plots present the action means at the corresponding time step, with the black contour depicting the $\alpha = 5$ level set. Note that positive mean turning velocity represents anticlockwise rotation. The halo above the agent is a visualization of its LiDAR observations for the hazard and goal. See Appendix E.5 for Case 1 (pre-trained π_{task}).



(a) (b)

Figure 3: Results from the reach-avoid experiment for both Case 1 (pre-trained π_{task}) and 2 (π_{task} trained using Projected PPO). (3a) Violation probabilities over different values of α. (3b) Mean and standard deviation episode length for successful trajectories for different values of α. Action seeding for each episode was controlled across different values of α and across the different cases, so all results depend on α and the training of π_{task}. The same figures but zoomed in to the scale across which $\epsilon_{\text{task}} \leq 1$ can be found in Appendix E.5.

These observations further confirm our earlier answers to Q1 and Q2, whilst now we also have an answer for Q3: the prior bound can be very conservative, though it is possible to see measurable improvement in performance while the bound remains fairly sensible.

9 Limits to Sporting SPoRt

The most obvious limitation of SPoRt is the conservativeness of the prior bound $\epsilon_{\text{task}} = \epsilon_{\text{base}}\alpha^T$, which prevents significant policy changes if the bound is to be used to guarantee safety. This conservativeness also results in the limitation of needing T to be as low as possible, making SPoRt unsuitable for applications where T is high (though we have seen ways to mitigate this limitation). Another limitations include the reliance on collecting many scenarios to obtain a useful bound ϵ_{base}, which may not be practical for some applications, as well as the requirement of stochastic policies (and, ideally, a fairly exploratory π_{base} to achieve noticeable policy change). Despite these theoretical limits, we have displayed the usefulness of the end-to-end architecture of SPoRt in meaningful simulation studies, which are promising for upcoming real-world implementations of SPoRt.

10 Conclusions

We have presented novel theoretical results that provide a prior bound on the probability of (safety) property violation for a task-specific policy in a model-free, episodic RL setup, based on a new 'maximum policy ratio' established vis-a-vis a given 'safe' base policy. Based on these bounds, we have presented an end-to-end architecture, SPoRt, which combines a data-driven approach for obtaining such a bound for the base policy with a projection-based approach for training the task-specific policy while maintaining a user-specified prior bound on (safety) property violation, thus trading off safety guarantees and task-specific performance. In view of promising experimental simulation results, future work will focus on reducing the conservativeness of the prior bound, to improve its utility in practical real-world applications.

Acknowledgments

This work was supported by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems [EP/S024050/1].

References

[Achiam *et al.*, 2017] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 06–11 Aug 2017.

[Agrawal *et al.*, 2018] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[Alshiekh *et al.*, 2018] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018.

[Altman, 2021] Eitan Altman. *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge, Boca Raton, 1 edition, December 2021.

[Calafiore and Campi, 2006] G.C. Calafiore and M.C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, May 2006.

[Campi and Garatti, 2008] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[Campi and Garatti, 2010] M. C. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: Feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2):257–280, October 2010.

[Campi and Garatti, 2018] Marco C Campi and Simone Garatti. *Introduction to the Scenario Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, November 2018.

[Chow *et al.*, 2018] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[Chu *et al.*, 2020] Tianshu Chu, Jie Wang, Lara Codeca, and Zhaojian Li. Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, March 2020.

[Diamond and Boyd, 2016] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018.

[Hasanbeig *et al.*, 2023] Hosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Certified reinforcement learning with logic guidance. *Artificial Intelligence*, 322:103949, September 2023.

[Hsu *et al.*, 2024] Kai-Chieh Hsu, Haimin Hu, and Jaime F. Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7(1):47–72, July 2024.

[Hwangbo *et al.*, 2019] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, January 2019.

[Isele *et al.*, 2018] David Isele, Alireza Nakhaei, and Kikuo Fujimura. Safe Reinforcement Learning on Autonomous Vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6, Madrid, October 2018. IEEE.

[Ji *et al.*, 2023] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[Kober and Peters, 2014] Jens Kober and Jan Peters. *Reinforcement Learning in Robotics: A Survey*, volume 97 of *Springer Tracts in Advanced Robotics*, page 9–67. Springer International Publishing, Cham, 2014.

[Konighofer *et al.*, 2023] Bettina Konighofer, Julian Rudolf, Alexander Palmisano, Martin Tappler, and Roderick Bloem. Online shielding for reinforcement learning. *Innovations in Systems and Software Engineering*, 19(4):379–394, December 2023.

[Lee *et al.*, 2023] Hyosun Lee, Yohee Han, and Youngchan Kim. Reinforcement learning for traffic signal control: Incorporating a virtual mesoscopic model for depicting oversaturated traffic conditions. *Engineering Applications of Artificial Intelligence*, 126:107005, November 2023.

[Li *et al.*, 2022] Guofa Li, Yifan Yang, Shen Li, Xingda Qu, Nengchao Lyu, and Shengbo Eben Li. Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness. *Transportation Research Part C: Emerging Technologies*, 134:103452, January 2022.

[Ma *et al.*, 2021] Xiaobai Ma, Jiachen Li, Mykel J. Kochenderfer, David Isele, and Kikuo Fujimura. Reinforcement Learning for Autonomous Driving with Latent State

Inference and Spatial-Temporal Relationships. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6064–6071, Xi'an, China, May 2021. IEEE.

[Mason and Grijalva, 2019] Karl Mason and Santiago Grijalva. A review of reinforcement learning for autonomous building energy management. *Computers & Electrical Engineering*, 78:300–312, 2019.

[Milosevic *et al.*, 2024] Nikola Milosevic, Johannes Müller, and Nico Scherf. Embedding Safety into RL: A New Take on Trust Region Methods. *arXiv*, November 2024.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, page 46–57, Providence, RI, USA, September 1977. IEEE.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

[Singh *et al.*, 2022] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990, February 2022.

[Soudjani and Abate, 2012] S. Esmaeil Zadeh Soudjani and A. Abate. Higher order approximations for verification of stochastic hybrid systems. In *Proceedings of ATVA12, LNCS 7561*, pages 416–434. Springer Verlag, 2012.

[Soudjani and Abate, 2013] Sadegh Esmaeil Zadeh Soudjani and Alessandro Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2):921–956, January 2013.

[Soudjani and Abate, 2015] Sadegh Esmaeil Zadeh Soudjani and Alessandro Abate. Quantitative approximation of the probability distribution of a markov process by formal abstractions. *Logical Methods in Computer Science*, Volume 11, Issue 3:1584, September 2015.

[Sutton and Barto, 2014] Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, nachdruck edition, 2014.

[Tempo *et al.*, 2005] R Tempo, Giuseppe Calafiore, and Fabrizio Dabbene. *Randomized algorithms for analysis and control of uncertain systems*. Communications and control engineering series. Springer, London, 2005.

[Vertovec and Margellos, 2023] Nikolaus Vertovec and Kostas Margellos. State Aggregation for Distributed Value Iteration in Dynamic Programming. *IEEE Control Systems Letters*, 7:2269–2274, 2023.

[Vertovec *et al.*, 2024] Nikolaus Vertovec, Kostas Margellos, and Maria Prandini. Finite sample learning of moving targets. *arXiv*, August 2024.

[Vidyasagar, 2003] M. Vidyasagar. *Learning and Generalisation*. Springer London, 2003.