

An Unreasonably Simple Approach to Safe RL

Geraud Nangue Tasse, Mark Nemecek, Tamlin Love, Steven James,
Benjamin Rosman

Keywords: Reinforcement Learning, Deep RL, Safe RL, Constrained MDP, Reward shaping

Summary

An important problem in reinforcement learning is designing agents that learn to solve tasks safely in an environment. A common solution is to define either a penalty in the reward function or a cost to be minimised when reaching unsafe states. However, designing reward or cost functions to incentivise safe behaviours with maximal probability is non-trivial and can increase with the complexity of the problem. To address this, we propose a simple algorithm for agents to automatically learn the penalty for unsafe transitions while learning to solve the given task. Precisely, the agent keeps track of its lowest and highest estimated state-values, then simply uses their difference as the penalty for unsafe transitions. Importantly, our approach can be used with any base RL algorithm since it only modifies the rewards perceived by an agent, and introduces no new hyperparameters that would require tuning. Our experiments demonstrate the effectiveness of this approach in enabling agents to learn safe policies in high-dimensional continuous control environments. Code: <https://github.com/mark-nemecek/rosarl>

Contribution(s)

1. We propose a parameter-free, model-agnostic algorithm for maximally safe RL that dynamically calculates safety penalties using the agent’s own value-function bounds.

Context: Traditional safe reinforcement learning often relies on complex Lagrangian multipliers, PID controllers, or manual reward shaping, all of which require extensive hyperparameter tuning and environment-specific knowledge (Amodei et al., 2016). Our approach, which sets the safety penalty as the difference between the estimated minimum and maximum state-values ($R_{\text{unsafe}} = V_{\text{MIN}} - V_{\text{MAX}}$), eliminates these overheads. By leveraging the agent’s internal expectations to scale the penalty, we provide a *just-sufficient* signal that enforces safety without the need for manual engineering or additional hyperparameters.

2. We demonstrate that the value-range penalty provides superior robustness to environmental stochasticity compared to state-of-the-art constrained RL baselines.

Context: Transition noise and environmental stochasticity frequently cause standard safe RL methods, such as TRPO-Lagrangian (Ray et al., 2019) or Sauté-TRPO (Sootla et al., 2022), to violate constraints as they struggle to balance reward maximization with unpredictable dynamics. We show that because our penalty is coupled with the current value range, it naturally *hardens* in response to the risks presented by the environment.

3. We empirically establish that simple reward augmentation can match or exceed the performance of complex constrained architectures in continual and high-dimensional continuous control tasks.

Context: Modern safety-critical RL research has trended toward increasingly complex, multi-component architectures (Ji et al., 2024). We challenge this trend by evaluating our simple reward-augmentation strategy on the Safety-Gymnasium suite, including PointGoal, PointPush, and Humanoid Velocity. We find that our unified penalty mechanism successfully enforces maximal safety across these diverse, high-dimensional benchmarks. This demonstrates that a simple, adaptive penalty is sufficient to regulate complex behaviors (such as humanoid locomotion) often outperforming more specialized safe RL algorithms.

An Unreasonably Simple Approach to Safe RL

Geraud Nangue Tasse^{1,4,†}, Mark Nemecek^{3,†}, Tamlin Love^{2,†}, Steven James^{1,4}, Benjamin Rosman^{1,4}

mark.nemecek@duke.edu, tlove@iri.upc.edu, {geraud.nanguetassel, steven.james, benjamin.rosman1}@wits.ac.za

¹School of Computer Science and Applied Mathematics, University of the Witwatersrand

²Institut de Robòtica i Informàtica Industrial, CSIC-UPC

³Department of Computer Science, Duke University

⁴Machine Intelligence and Neural Discovery Institute, University of the Witwatersrand

[†] Equal contribution

Abstract

An important problem in reinforcement learning is designing agents that learn to solve tasks safely in an environment. A common solution is to define either a penalty in the reward function or a cost to be minimised when reaching unsafe states. However, designing reward or cost functions to incentivise safe behaviours with maximal probability is non-trivial and can increase with the complexity of the problem. To address this, we propose a simple algorithm for agents to automatically learn the penalty for unsafe transitions while learning to solve the given task. Precisely, the agent keeps track of its lowest and highest estimated state-values, then simply uses their difference as the penalty for unsafe transitions. Importantly, our approach can be used with any base RL algorithm since it only modifies the rewards perceived by an agent, and introduces no new hyperparameters that would require tuning. Our experiments demonstrate the effectiveness of this approach in enabling agents to learn safe policies in high-dimensional continuous control environments. Code: <https://github.com/mark-nemecek/rosarl>

1 Introduction

Reinforcement learning (RL) has recently achieved success across a variety of domains, such as video games (Shao et al., 2019), robotics (Kalashnikov et al., 2018; Kahn et al., 2018) and autonomous driving (Kiran et al., 2021). However, if we hope to deploy RL in the real world, agents must be



Figure 1: Sample trajectories of representative prior works—TRPO (Schulman et al., 2015) (left-most), TRPO-Lagrangian (Ray et al., 2019) (middle-left), and CPO (Achiam et al., 2017) (middle-right)—compared to ours (right-most) in Safety Gym (Ray et al., 2019). The TRPO-ValueRange trajectory illustrates our safety-first objective: reducing the probability of safety violations by taking the longer path when sensor observations are limited, rather than just satisfying an expected-cost budget.

capable of completing tasks while avoiding unsafe or costly behaviour. For example, a navigating robot must avoid colliding with objects and actors around it, while simultaneously learning to solve the required task. Figure 1 shows an example.

Many approaches in RL deal with this problem by allocating arbitrary penalties to unsafe states when hand-crafting the reward function. However, the problem of specifying a reward function for desirable, safe behaviour is notoriously difficult (Amodi et al., 2016). *Importantly, penalties that are too small may result in unsafe behaviour, while penalties that are too large may result in increased learning times.* Furthermore, these rewards must be specified by an expert for each new task an agent faces. If our aim is to design truly autonomous, general agents, it is then simply impractical to require that a human designer specify penalties to guarantee optimal but safe behaviours for every task.

When safety is an explicit goal, a common approach is to constrain policy learning according to some threshold on cumulative cost (Schulman et al., 2015; Ray et al., 2019; Achiam et al., 2017). While effective, these approaches require either sparse costs or the design of a dense cost function whose specification can be as challenging as designing a reward function. This is especially difficult in our setting of interest where feasible policies must achieve the lowest possible number of violations. Additionally, these methods may still result in unacceptably frequent constraint violations in practice (even with sparse costs), due to the large cost threshold typically used (for example, 25 in Ray et al. (2019)).

Rather than attempting to both maximise a reward function and minimise a cost function, which requires specifying both rewards and costs and a new learning objective, we should simply aim to have a better reward function—since we then do not have to specify yet another scalar signal nor change the learning objective. This approach is consistent with the *reward hypothesis* (Sutton & Barto, 2018) which states: “*All of what we mean by goals and purposes can be well thought of as maximisation of the expected value of the cumulative sum of a received scalar signal (reward).*” Therefore, the question we examine in this work is how to determine a sufficient penalty for unsafe transitions such that an optimal agent prioritises *maximal safety*: minimising the probability of safety violations, and achieving *almost-sure* safety whenever probability 1 is attainable. Rather than requiring an expert’s input like prior works, we show that this penalty can be learned autonomously by an agent using its current value estimates.

2 Background

Safe RL is often modelled by a constrained Markov Decision Process (CMDP) (Altman, 1999). A CMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, C, d, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition probability function, $R : \mathcal{S} \times \mathcal{A} \rightarrow [R_{\min}, R_{\max}]$ is the reward function, $C : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the cost function, $d \in \mathbb{R}$ is the cost threshold, and $\gamma \in (0, 1)$ is the discount factor. The agent is then associated with a *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which it uses to take actions in the environment. This policy can then be characterised by its cost value function $V_C^\pi(s) = \mathbb{E}^\pi[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t)]$, and the policy is *safe* if $V_C^\pi(s) - d \leq 0$ for all $s \in \mathcal{S}$. Where Π is the set of all safe policies, the goal of an agent is then to learn an optimal safe policy π^* that maximises the value function $V^{\pi^*}(s) = \max_{\pi \in \Pi} V^\pi(s)$ for all $s \in \mathcal{S}$ (Ray et al., 2019).

Maximally Safe RL (our problem setting): In contrast to most prior works which formulate safety as a constraint on *expected* cumulative cost, we consider safety-critical settings where the primary objective is to maximise the probability of satisfying the safety condition. We call a policy *maximally safe* if it achieves the greatest safety probability attainable in the environment. When this probability is 1, maximal safety is exactly almost-sure safety (Sootla et al., 2022). When unavoidable stochasticity makes probability 1 impossible, the objective remains well-defined—the agent should be as safe as the environment permits. Hence, following the formulation of Sootla et al. (2022), we consider a safety-augmented MDP $\widehat{\mathcal{M}}$ as a tuple $\langle \widehat{\mathcal{S}}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$ where $\widehat{\mathcal{S}} = \mathcal{S} \times \mathcal{Z}$ and $\mathcal{Z}$ is a set of safety indicators, $\widehat{P} : \widehat{\mathcal{S}} \times \mathcal{A} \times \widehat{\mathcal{S}} \rightarrow [0, 1]$ defines the transitions from the current state $s \in \mathcal{S}$ and safety

indicator $z \in \mathcal{Z}$ after each action $a \in \mathcal{A}$, and $\widehat{R}((s, z), a) = \begin{cases} R(s, a) & \text{if } z \leq 0, \\ R_{\text{unsafe}} & \text{otherwise,} \end{cases}$ is the reshaped

reward function where $R_{\text{unsafe}} \in [-\infty, 0)$ penalises unsafe transitions. In essence, the higher z is, the more unsafe the current transition is (for example, z can be proportional to the episodic cumulative cost). This formulation enables us to use standard RL to solve CMDPs, assuming an appropriate choice for $\mathcal{Z}$ and R_{unsafe} . For example, by using Sauté MDPs (Sootla et al., 2022) with $R_{\text{unsafe}} = -\infty$. However, the question of what finite penalty R_{unsafe} to use in practice to obtain safe optimal policies remains open. To investigate this question, we retrain our attention to safety critical settings where the environment terminates immediately when the safe condition is violated ($z > 0$).

3 Learning safe policies

The primary challenge in maximally safe RL is determining a finite penalty R_{unsafe} that is sufficiently large to discourage unsafe transitions without being so excessive that it destabilises the learning process or leads to exploding gradients. We propose a dynamic approach that leverages the agent’s own evolving perception of the environment’s value range.

3.1 The Value-Range Penalty

Consider the LAVA GRIDWORLD (Russell & Norvig, 2016) as a running example in Figure 2: the agent must reach goal G while avoiding lava L , which causes an immediate safety violation and termination (more details in Appendix 7). Stochasticity here is controlled by a uniform *slip probability* (sp). In the deterministic setting ($sp = 0$), the optimal safe path can closely border the lava. However as the slip probability increases, a longer route can have substantially lower failure probability. Thus, a maximally safe agent should increasingly prefer the longer route as stochasticity increases.

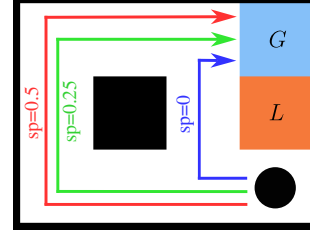


Figure 2: LAVA GRIDWORLD

This example also illustrates why using only $R_{\text{unsafe}} = -V_{\text{MAX}}$ is insufficient. For example, when $-V_{\text{MAX}} = -R_{\text{MAX}} = -1$, the agent can prefer the shorter, unsafe route when $sp = 0.5$. For reward structures such as $R_{\text{MAX}} = 0$ or a step reward of -1 , such a penalty can even lead to the lava across all sp . In general, the agent could face a worst-case scenario where from a given state, an unsafe behaviour might appear to offer the highest possible cumulative return, while the safest path offers the lowest possible cumulative return. A penalty must therefore account not only for the best attainable values but also for the full range of outcomes that the agent currently perceives.

We address this by defining the penalty based on the range of the value function. Let $V(s)$ be the estimated value of state s . We maintain running estimates of the minimum and maximum values encountered during training, V_{MIN} and V_{MAX} , and define $R_{\text{unsafe}} = V_{\text{MIN}} - V_{\text{MAX}}$. This choice of R_{unsafe} is grounded in the intuition of regret minimization. By setting the penalty to the negative of the value gap, we ensure that the *cost* of failing is always greater than the potential *gain* of any sequence of rewards within the environment’s normal operating range. Because V_{MIN} and V_{MAX} are updated online, the penalty R_{unsafe} starts small (when the agent is uncertain) and grows in magnitude as the agent discovers higher-reward regions, maintaining a consistent pressure to prioritize safety.

3.2 Algorithm and Implementation

The proposed method, detailed in Algorithm 1, integrates seamlessly with any model-free RL algorithm. It requires no prior knowledge of the environment’s reward bounds and introduces no additional hyperparameters. The agent tracks the reward bounds R_{MIN} , R_{MAX} and the value bounds V_{MIN} , V_{MAX} at each timestep. Crucially, V_{MIN} and V_{MAX} are updated using both the observed rewards and the current value function $V(s_t)$, ensuring that the penalty reflects the agent’s current long-term expectations. When a safety violation occurs ($z_t > 0$), the environment’s reward r_t is replaced by the current R_{unsafe} before the RL update occurs. Since V_{MAX} is estimated optimistically through $\max(V_{\text{MAX}}, r_t, V(s_t))$, the resulting penalty R_{unsafe} is naturally *conservative* enough to enforce safety.

Algorithm 1: Safe RL with a learned unsafe penalty

Input : RL algorithm **A**, max timesteps T
Initialise : $R_{\text{MIN}} = 0, R_{\text{MAX}} = 0, V_{\text{MIN}} = R_{\text{MIN}}, V_{\text{MAX}} = R_{\text{MAX}}, \pi$ and V as per **A**
for $t \in T$ **do**
 observe a state s_t , **take** an action a_t using π as per **A**
 observe the next state s_{t+1} , reward r_t , and the safety indicator z_t
 $R_{\text{MIN}}, R_{\text{MAX}} \leftarrow \min(R_{\text{MIN}}, r_t), \max(R_{\text{MAX}}, r_t)$
 $V_{\text{MIN}}, V_{\text{MAX}} \leftarrow \min(V_{\text{MIN}}, R_{\text{MIN}}, V(s_t)), \max(V_{\text{MAX}}, R_{\text{MAX}}, V(s_t))$
 $R_{\text{unsafe}} \leftarrow V_{\text{MIN}} - V_{\text{MAX}}$
 $r_t \leftarrow R_{\text{unsafe}}$ **if** $z_t > 0$ **else** r_t
 update π and V with (s_t, a_t, s_{t+1}, r_t) as per **A**
end for

4 Experiments

To validate our approach, We first use the LAVA GRIDWORLD example to isolate the behaviour of the learned penalty under controlled stochasticity, before comparing with representative baselines in challenging continuous-control domains requiring function approximation.

4.1 Q-Learning Instantiation

We instantiate Algorithm 1 with Q-learning (Watkins, 1989) in the LAVA GRIDWORLD domain of Figure 2. We use ϵ -greedy exploration with $\epsilon = 0.1$, learning rate $\alpha = 0.1$, and initialise all action-values to 0. The ValueRange penalty is updated online and replaces the reward whenever the lava state is reached.

We vary environmental stochasticity through the slip probability sp . At $sp = 0$, the agent can safely take the shortest path bordering the lava; as sp increases, maximising safety probability requires increasingly conservative routes. Figures 3a–c compare our adaptive penalty with a deliberately large fixed penalty, $R_{\text{unsafe}} = -10000$. Both can learn to reduce failures, but the excessively large fixed penalty struggles to maximise reward under stochasticity. In contrast, ValueRange adapts its scale online, maintaining low failure rates while better preserving task-return optimisation. Figure 3d further isolates the effect of penalty magnitude by sweeping a fixed R_{unsafe} from -1 to -10^6 . As the penalty becomes increasingly negative, the agent progressively struggles to maximise reward and learning variance increases. Importantly, this degradation is not monotonic in environmental stochasticity: for sufficiently large penalties, $sp = 0.25$ attains worse returns than the more stochastic $sp = 0.5$. Thus, simply making the unsafe penalty arbitrarily large does not monotonically improve learning; an excessive penalty can dominate reward differences between behaviours and make learning good safe policies increasingly difficult. This motivates adapting the penalty to the task’s value scale rather than manually selecting an extremely conservative constant.

4.2 Continuous-control evaluation

We next investigate the empirical performance of our proposed penalty algorithm across a variety of challenging continuous-control tasks (Figure 4). We aim to address four primary research questions: (i) The effect of environmental stochasticity on safety. (ii) Performance under episodic goal constraints. (iii) Survival and success in continual goal settings. (iv) The ability to regulate high-dimensional control without explicit spatial goals. Each method is trained over 10 independent random seeds. For post-training evaluation, each of the 10 trained models is evaluated over 100 random episodes, giving 1000 evaluation rollouts per method and task. Tables report the mean $\pm$ one standard deviation.

Baselines. We consider representative OmniSafe baselines (Ji et al., 2024) in various Safety-gymnasium domains (Ji et al., 2023), including the PILLAR domain. As a baseline representative of

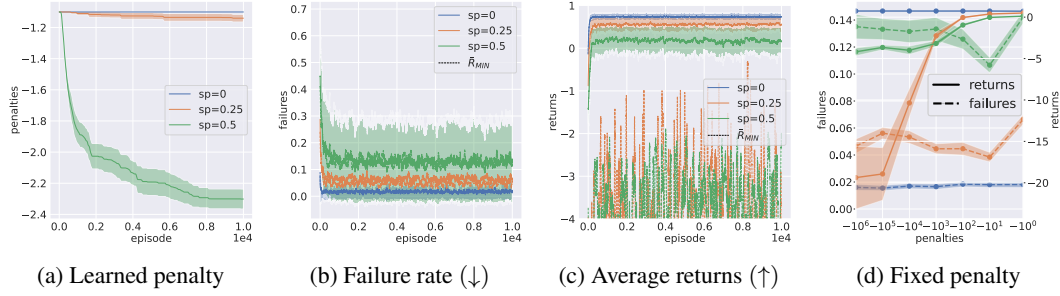


Figure 3: LAVA GRIDWORLD results. (a–c) Effect of increasing slip probability sp on the learned ValueRange penalty, failure rate, and return, compared with a fixed large penalty $\bar{R}_{MIN} = -10000$. ValueRange adapts its penalty to environmental stochasticity while maintaining low failure rates and better preserving task performance. (d) Sweeping a fixed penalty from -1 to -10^6 shows that increasingly large penalties make reward maximisation harder and increase learning variance. Notably, at large penalty magnitudes, $sp = 0.25$ can achieve worse returns than the more stochastic $sp = 0.5$. These results illustrate that “larger” is not necessarily “better” for an unsafe-transition penalty and motivate automatically adapting its scale. Results are averaged over 20 random seeds and shaded regions denote one standard deviation.

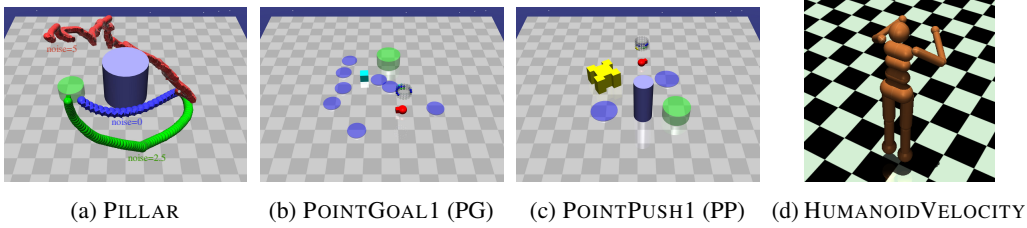


Figure 4: Safety-Gym(nasium) domains (Ray et al., 2019; Ji et al., 2024). The agent here must: (a) Navigate (in $|\mathcal{S}| = \mathbb{R}^{60}$) to the green cylinder while avoiding the pillar. See Appendix 7 for detailed descriptions. (b–d) These are standard Safety-Gymnasium tasks, only modified to terminate when an unsafe state is reached: entering a blue region in POINTGOAL1 ($|\mathcal{S}| = \mathbb{R}^{60}$) and exceeding the velocity threshold in HUMANOIDVELOCITY ($|\mathcal{S}| = \mathbb{R}^{348}$).

typical RL approaches, we use Trust Region Policy Optimisation (TRPO) (Schulman et al., 2015). To represent constraint-based approaches and PID Lagrangian methods, we compare against, TRPO with Lagrangian constraints (TRPO-Lagrangian) (Ray et al., 2019), Sauté RL with TRPO (Sauté-TRPO) (Sootla et al., 2022), Constrained Proximal Policy Optimisation with PID (CPPPID) (Stooke et al., 2020), and Penalised Proximal Policy Optimisation (P3O) (Zhang et al., 2022). All baselines use the implementations provided by Ji et al. (2024), and form a set of widely used baselines in safety domains (Zhang et al., 2020; Sootla et al., 2022; Yang et al., 2023; Ji et al., 2024). We use the same hyperparameters as in Ji et al. (2024), such as feed-forward MLPs for all approaches with value networks of size (256,256) and $\tanh$ activation functions, except for a cost threshold of 0 (which gave the best baseline results compared to 1 and the default of 25). **Ours** is TRPO-ValueRange.

Robustness to Environmental Stochasticity To evaluate robustness to stochasticity, we vary the transition noise in PILLAR (Table 1). In the deterministic setting, most methods perform competitively and achieve zero mean cost. As stochasticity increases, TRPO-ValueRange consistently achieves the lowest mean cost and highest mean success rate, while several expectation-constrained and unconstrained baselines exhibit increasing violations. These results support the intended behaviour of the adaptive penalty: as perfect safety becomes harder or impossible, the learned policy increasingly favours behaviours with greater achievable safety probability.

Noise	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps
0.0	Ours	0.00 ± 0.00	1.00 ± 0.00	3.21 ± 0.00	136.60 ± 12.32
	TRPO-Lagrangian	0.00 ± 0.00	1.00 ± 0.00	3.21 ± 0.00	137.52 ± 14.50
	Sauté-TRPO	0.00 ± 0.00	0.99 ± 0.02	3.20 ± 0.03	142.88 ± 12.37
	TRPO	0.00 ± 0.00	1.00 ± 0.00	3.21 ± 0.00	138.92 ± 14.47
	CPPOPID	0.00 ± 0.00	1.00 ± 0.00	3.21 ± 0.00	140.70 ± 20.65
	P3O	0.10 ± 0.30	0.83 ± 0.35	2.74 ± 1.01	205.50 ± 220.31
1.5	Ours	0.06 ± 0.02	0.94 ± 0.02	3.01 ± 0.08	262.19 ± 28.06
	TRPO-Lagrangian	0.09 ± 0.04	0.91 ± 0.04	2.90 ± 0.12	255.55 ± 26.62
	Sauté-TRPO	0.11 ± 0.04	0.89 ± 0.04	2.81 ± 0.14	232.26 ± 10.55
	TRPO	0.13 ± 0.08	0.87 ± 0.08	2.74 ± 0.29	262.91 ± 32.70
	CPPOPID	0.17 ± 0.08	0.83 ± 0.08	2.61 ± 0.31	344.60 ± 65.83
	P3O	0.11 ± 0.13	0.76 ± 0.33	2.43 ± 1.04	391.09 ± 221.08
2.5	Ours	0.14 ± 0.05	0.80 ± 0.11	2.61 ± 0.27	503.49 ± 98.67
	TRPO-Lagrangian	0.20 ± 0.05	0.72 ± 0.24	2.38 ± 0.46	461.89 ± 132.78
	Sauté-TRPO	0.19 ± 0.09	0.76 ± 0.24	2.45 ± 0.54	435.18 ± 104.06
	TRPO	0.28 ± 0.10	0.63 ± 0.22	2.05 ± 0.52	446.21 ± 143.94
	CPPOPID	0.28 ± 0.16	0.66 ± 0.22	2.12 ± 0.64	485.50 ± 76.94
	P3O	0.17 ± 0.07	0.71 ± 0.16	2.42 ± 0.34	552.42 ± 139.28

Table 1: Performance in PILLAR under increasing environmental noise. TRPO-ValueRange has the lowest mean cost at every noise level (tied at zero in the deterministic setting) while retaining the highest mean success rate under stochasticity, illustrating the maximal-safety behaviour targeted by our objective.

Task	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps
PG	Ours	0.04 ± 0.03	0.50 ± 0.17	0.84 ± 0.45	532.08 ± 148.18
	TRPO-Lagrangian	0.09 ± 0.03	0.86 ± 0.03	1.76 ± 0.08	119.18 ± 19.82
	Sauté-TRPO	0.12 ± 0.03	0.87 ± 0.03	1.77 ± 0.09	77.97 ± 10.33
	TRPO	0.10 ± 0.02	0.90 ± 0.02	1.84 ± 0.04	73.86 ± 4.37
	CPPOPID	0.11 ± 0.03	0.89 ± 0.03	1.81 ± 0.08	73.65 ± 5.04
	P3O	0.08 ± 0.02	0.91 ± 0.02	1.86 ± 0.06	101.98 ± 13.03
PP	Ours	0.08 ± 0.03	0.01 ± 0.01	0.47 ± 0.11	940.58 ± 20.33
	TRPO-Lagrangian	0.12 ± 0.05	0.03 ± 0.03	0.62 ± 0.21	914.53 ± 29.49
	Sauté-TRPO	0.13 ± 0.05	0.08 ± 0.14	0.92 ± 0.73	905.51 ± 33.87
	TRPO	0.14 ± 0.06	0.05 ± 0.06	0.72 ± 0.37	903.23 ± 37.82
	CPPOPID	0.13 ± 0.04	0.04 ± 0.03	0.52 ± 0.11	889.25 ± 38.75
	P3O	0.13 ± 0.04	0.06 ± 0.05	0.76 ± 0.33	921.17 ± 26.78

Table 2: Results for episodic goal-reaching tasks. TRPO-ValueRange achieves the lowest mean cost in both PG and PP, illustrating its safety-first trade-off: lower violation probability is prioritised even when this requires sacrificing shorter, higher-return behaviour.

Safety Trade-offs in Episodic Goal Tasks In episodic navigation tasks such as POINTGOAL1 (PG) and POINTPUSH1 (PP), TRPO-ValueRange consistently achieves the lowest mean cost among the reported baselines (Table 2). This safety-first behaviour can reduce success rate or return relative to methods that accept greater violation risk, reflecting the distinction between our maximal-safety objective and approaches that optimise reward subject to an expected-cost budget. When probability-1 safety is feasible this preference corresponds to almost-sure safety, otherwise it favours the safest behaviour achievable in the environment.

Task	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps ↑
PG	Ours	0.08 ± 0.05	0.45 ± 0.16	1.87 ± 1.60	950.31 ± 31.45
	TRPO-Lagrangian	0.62 ± 0.07	0.34 ± 0.06	10.17 ± 0.77	607.44 ± 56.15
	Sauté-TRPO	0.79 ± 0.03	0.21 ± 0.03	11.01 ± 0.55	493.01 ± 24.51
	TRPO	0.78 ± 0.05	0.22 ± 0.05	10.68 ± 0.74	483.42 ± 33.07
	CPPPOPID	0.80 ± 0.04	0.20 ± 0.03	10.31 ± 0.88	482.05 ± 36.60
	P3O	0.56 ± 0.07	0.42 ± 0.07	10.63 ± 0.74	667.08 ± 49.14
PP	Ours	0.09 ± 0.03	0.05 ± 0.04	0.53 ± 0.15	905.26 ± 20.12
	TRPO-Lagrangian	0.11 ± 0.03	0.13 ± 0.18	0.83 ± 0.49	844.21 ± 110.39
	Sauté-TRPO	0.12 ± 0.05	0.10 ± 0.12	0.68 ± 0.30	838.98 ± 106.42
	TRPO	0.15 ± 0.07	0.16 ± 0.21	0.86 ± 0.56	795.70 ± 157.18
	CPPPOPID	0.16 ± 0.03	0.04 ± 0.01	0.54 ± 0.09	896.50 ± 19.31
	P3O	0.13 ± 0.06	0.11 ± 0.12	0.78 ± 0.36	859.75 ± 74.76

Table 3: Performance in continual goal tasks. In PG, TRPO-ValueRange combines the lowest mean cost with the highest mean success rate and longest survival, showing that its high Total Steps arise from longer, safer goal-directed behaviour rather than inactivity.

Survival and Success in Continual Goal Settings Continual goal tasks require agents to repeatedly pursue objectives without resetting, so survival must be interpreted jointly with task success. In POINTGOAL1, TRPO-ValueRange achieves a mean cost of 0.08, survives for 950 of the maximum 1000 evaluation steps, *and attains the highest mean success rate* (0.45) among the reported baselines (Table 3). Thus, its longer survival and lower return are not explained by simply remaining stationary: the agent continues reaching goals, but preferentially takes longer and safer routes when possible. This is precisely the qualitative behaviour expected from a maximal-safe agent.

Safety without explicit goals Finally, we evaluate the high-dimensional HUMANOIDVELOCITY task, where the agent must maintain a high walking speed without exceeding a safety threshold. TRPO-ValueRange achieves the lowest mean cost (0.01) among the reported methods (Table 4). Methods such as Sauté-TRPO obtain higher returns but incur substantially more violations, again exposing the safety–reward trade-off. These results show that the value-range penalty can regulate safety even in a high-dimensional continuous-control problem without an explicit spatial goal.

Task	Costs ↓	Returns ↑	Total Steps ↑
Ours	0.01 ± 0.01	610.50 ± 53.62	119.61 ± 10.06
TRPO-Lagrangian	0.09 ± 0.04	800.09 ± 100.72	147.62 ± 18.21
Sauté-TRPO	0.35 ± 0.12	957.11 ± 165.28	173.76 ± 28.66
TRPO	0.43 ± 0.18	944.04 ± 206.16	172.22 ± 35.41
CPPPOPID	0.19 ± 0.06	805.04 ± 191.88	150.88 ± 33.30
P3O	0.02 ± 0.06	611.15 ± 42.52	121.90 ± 8.98

Table 4: Comparison in the HUMANOIDVELOCITY task. Our approach provides the most consistent velocity regulation with minimal safety violations.

5 Related Work

Reward shaping. The problem of designing reward functions to induce desired policies is well studied (Singh et al., 2009). Classical reward shaping augments an existing reward function to accelerate learning while preserving its optimal policy (Ng et al., 1999; Devidze et al., 2021). Other approaches explicitly shape rewards for safety: for example, Lipton et al. (2016) learn an intrinsic-fear model predicting near-future unsafe states and penalise Q-learning targets accordingly. Several works

are particularly related to our use of terminal penalties. [Thomas et al. \(2021\)](#) derive a sufficient safety-violation penalty using empirical reward extrema and a safety-horizon parameter H , within a model-based approach that predicts unsafe future trajectories. In a different deterministic goal-reaching setting, [Nangue Tasse et al. \(2020; 2025\)](#) assign unintended terminal goals a fixed reward satisfying $\bar{R}_{\min} \leq \min\{R_{\min}, (R_{\min} - R_{\max})D\}$, where D is the MDP diameter, ensuring that optimal policies achieve intended goals almost-surely. [Sun et al. \(2021\)](#) formulate Early-Terminated MDPs, augmenting the state with safety-budget information for cumulative constraints and using an absorbing terminal reward. They then show that a sufficiently adverse terminal reward preserves the corresponding safe optimal policies. These works demonstrate the usefulness of sufficiently adverse terminal rewards, but require quantities or choices such as a safety horizon H , environment diameter D , or a task-dependent terminal penalty. In contrast, We learn the penalty online directly from the agent’s evolving value span, requiring none of these additional quantities or penalty hyperparameters.

Constrained RL. Disincentivising or preventing undesirable behaviours is core to the field of safe RL. A popular approach is to define constraints on the behaviour of an agent using CMDPs, tasking the agent with limiting the accumulation of costs associated with violating safety constraints while simultaneously maximising reward ([Altman, 1999](#); [Achiam et al., 2017](#); [Chow et al., 2018](#); [Ray et al., 2019](#); [HasanzadeZonuzu et al., 2021](#)). Widely used examples of these approaches include constrained policy optimisation (CPO) ([Achiam et al., 2017](#)), which augments TRPO ([Schulman et al., 2015](#)) with constraints to satisfy a constrained MDP, and TRPO-Lagrangian ([Ray et al., 2019](#)), which combines Lagrangian methods with TRPO. Another example is Sauté RL ([Sootla et al., 2022](#)), which incorporates the cost function into the rewards and augments the state with the remaining "cost budget" spent by violating safety constraints. Other constraint-based approaches include Projection-based CPO ([Yang et al., 2020](#)), which projects a TRPO policy onto a space defined by constraints, and PID Lagrangian methods ([Stooke et al., 2020](#)), which augment Lagrangian methods with PID control. In contrast to most of the prior works which seek to maximise reward subject to a constraint on expected cumulative cost, our target is *maximal safety probability*: almost-sure safety when achievable (similarly to [Sootla et al. \(2022\)](#)), and otherwise the greatest safety probability attainable in the environment.

Shielding. This is an important line of work which relies on interventions from a model ([Dalal et al., 2018](#); [Wagener et al., 2021](#)) or human ([Tennenholtz et al., 2022](#)) to prevent unsafe actions from being considered by the agent (shielding the agent) or prevent the environment from executing those unsafe actions by correcting them (shielding the environment). Other approaches here also look at using temporal logics to define or enforce safety constraints on the actions considered or selected by the agent ([Alshiekh et al., 2018](#)). These approaches fit seamlessly into our reward-only framework since they are primarily about modifications on the transition dynamics and not the reward function—for example, in identifying unsafe actions that should immediately lead to termination with positive cost.

6 Conclusion

We studied safe reinforcement learning through the lens of a deliberately simple question: *can a single scalar reward signal be sufficient to learn policies that behave safely?* Focusing on safety-critical settings where violations terminate an episode, we define the target as maximising the probability of satisfying safety constraints—recovering almost-sure safety whenever probability 1 is achievable, while still specifying the safest attainable behaviour when unavoidable stochasticity makes perfect safety impossible. To achieve this, we introduced a reward-only approach that learns the unsafe-transition penalty directly from the agent’s own value estimates by tracking empirical lower and upper bounds and setting R_{unsafe} to their difference. Our results demonstrate consistent prioritisation of safety and typically achieves lower violation rates than widely used constraint-based and Lagrangian baselines, while remaining competitive in task performance when safety permitted. Practically, the method is plug-and-play: it applies to any off-the-shelf RL algorithm that maintains value estimates, introduces no additional hyperparameters, and avoids manual reward and cost tuning. Overall, these results suggest that autonomously learning a principled safety penalty within the reward function is a simple yet effective path toward more independent agents that can learn to act safely in complex environments.

Acknowledgments

Computations were performed using the High Performance Computing Infrastructure provided by the Mathematical Sciences Support unit at the University of the Witwatersrand and that provided by the Department of Computer Science at Duke University. B.R. is a CIFAR Azrieli Global Scholar in the Learning in Machines & Brains program. M.N. thanks Ron Parr for his support and sponsorship. T. L. was supported by the Horizon Europe Marie Skłodowska-Curie grant agreement No. 101072488 (TRAIL).

References

- Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.
- Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Eitan Altman. *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.
- Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.
- Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-based approach to safe reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.
- Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.
- Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. Explicable reward design for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34: 20118–20131, 2021.
- Aria HasanzadeZonuzi, Archana Bura, Dileep Kalathil, and Srinivas Shakkottai. Learning with safety constraints: Sample complexity of reinforcement learning for constrained MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7667–7674, 2021.
- 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. URL <https://openreview.net/forum?id=WZmlxIuIGR>.
- Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *Journal of Machine Learning Research*, 25(285):1–6, 2024.
- Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5129–5136. IEEE, 2018.
- Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018.
- B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

- Zachary C Lipton, Kamyar Azizzadenesheli, Abhishek Kumar, Lihong Li, Jianfeng Gao, and Li Deng. Combating reinforcement learning’s Sisyphian curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*, 2016.
- Geraud Nangue Tasse, Steven James, and Benjamin Rosman. A Boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9497–9507, 2020.
- Geraud Nangue Tasse, Steven James, and Benjamin Rosman. Composition and zero-shot transfer with lattice structures in reinforcement learning. *Journal of Artificial Intelligence Research*, 82: 2325–2388, 2025.
- Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pp. 278–287, 1999.
- Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.
- Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897. PMLR, 2015.
- Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*, 2019.
- Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from? In *Proceedings of the Annual Conference of the Cognitive Science Society*, pp. 2601–2606. Cognitive Science Society, 2009.
- Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyang Wang, David H Mguni, Jun Wang, and Haitham Ammar. Sauté RL: Almost surely safe reinforcement learning using state augmentation. In *International Conference on Machine Learning*, pp. 20423–20443. PMLR, 2022.
- Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID Lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.
- Hao Sun, Ziping Xu, Meng Fang, Zhenghao Peng, Jiadong Guo, Bo Dai, and Bolei Zhou. Safe exploration by solving early terminated mdp. *arXiv preprint arXiv:2107.04200*, 2021.
- Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Guy Tennenholtz, Nadav Merlis, Lior Shani, Shie Mannor, Uri Shalit, Gal Chechik, Assaf Hallak, and Gal Dalal. Reinforcement learning with a terminator. *Advances in Neural Information Processing Systems*, 35:35696–35709, 2022.
- Garrett Thomas, Yuping Luo, and Tengyu Ma. Safe reinforcement learning by imagining the near future. *Advances in Neural Information Processing Systems*, 34:13859–13869, 2021.
- Nolan C Wagener, Byron Boots, and Ching-An Cheng. Safe reinforcement learning using advantage-based intervention. In *International Conference on Machine Learning*, pp. 10630–10640. PMLR, 2021.
- C. Watkins. *Learning from delayed rewards*. PhD thesis, King’s College, Cambridge, 1989.
- Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

Yujie Yang, Yuxuan Jiang, Yichen Liu, Jianyu Chen, and Shengbo Eben Li. Model-free safe reinforcement learning through neural barrier certificate. *IEEE Robotics and Automation Letters*, 8(3):1295–1302, 2023.

Linrui Zhang, Li Shen, Long Yang, Shixiang Chen, Bo Yuan, Xueqian Wang, and Dacheng Tao. Penalized proximal policy optimization for safe reinforcement learning. *arXiv preprint arXiv:2205.11814*, 2022.

Yiming Zhang, Quan Vuong, and Keith Ross. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

7 Experiments Domains

LAVA GRIDWORLD This is a gridworld with 11 positions ($|\mathcal{S}| = 11$) and 4 cardinal actions ($|\mathcal{A}| = 4$). See Figure 2. The agent here must reach a goal location G while avoiding a lava location L (both terminal). A wall is also present in the environment and, while not unsafe, must be navigated around. The environment has a *slip probability* (sp), so that with probability sp the agent’s action is overridden with a random action. The agent receives $R_{\text{MAX}} = +1$ reward for reaching the goal, as well as $R_{\text{step}} = -0.1$ reward at each timestep to incentivise taking the shortest path to the goal.

Safety Gym PILLAR This is a custom Safety Gym domain in which the simple point robot must navigate to a goal location  around a large pillar  (both terminal). See Figure 18. All details of the environment are the same as in Ray et al. (2019) except when stated otherwise. Just as in Ray et al. (2019), the agent uses *pseudo-lidar* to observe the distance to objects around it ($|\mathcal{S}| = \mathbb{R}^{60}$), and the action space is continuous over two actuators controlling the direction and forward velocity ($|\mathcal{A}| = [-1, 1]^2$). This direction and forward velocity can be noisy, determined by a *noise* scalar as follows: $a_{\text{new}} = a + (\text{noise})a_{\text{noise}}$ where a_{new} is the new direction and forward velocity, $a \in \mathcal{A}$ is the agent’s action, and $a_{\text{noise}} \in \mathcal{A}$ is a uniformly sampled random vector. The goal, pillar, and agent locations remain unchanged for all episodes. Each episode terminates once the agent reaches the goal or collides with the pillar (with a reward of -1). Otherwise, episodes terminate after 1000 timesteps.

Safety Gymnasium We use these to compare the algorithms in the same default tasks as prior works. In the POINTGOAL1 task (Figure 4b), a simple robot must navigate to a goal location  across a 2D plane while avoiding several hazards . The agent’s sensors, actions, and rewards are identical to the PILLAR domain. Unlike the PILLAR domain, the goal location is randomly reset when the agent reaches it, but does not terminate the episode. The POINTGOAL1 task (Figure 4c) is similar to POINTGOAL1, but with the addition of a pillar obstacle  and a large box  the agent must push to the goal location  to receive the goal reward.

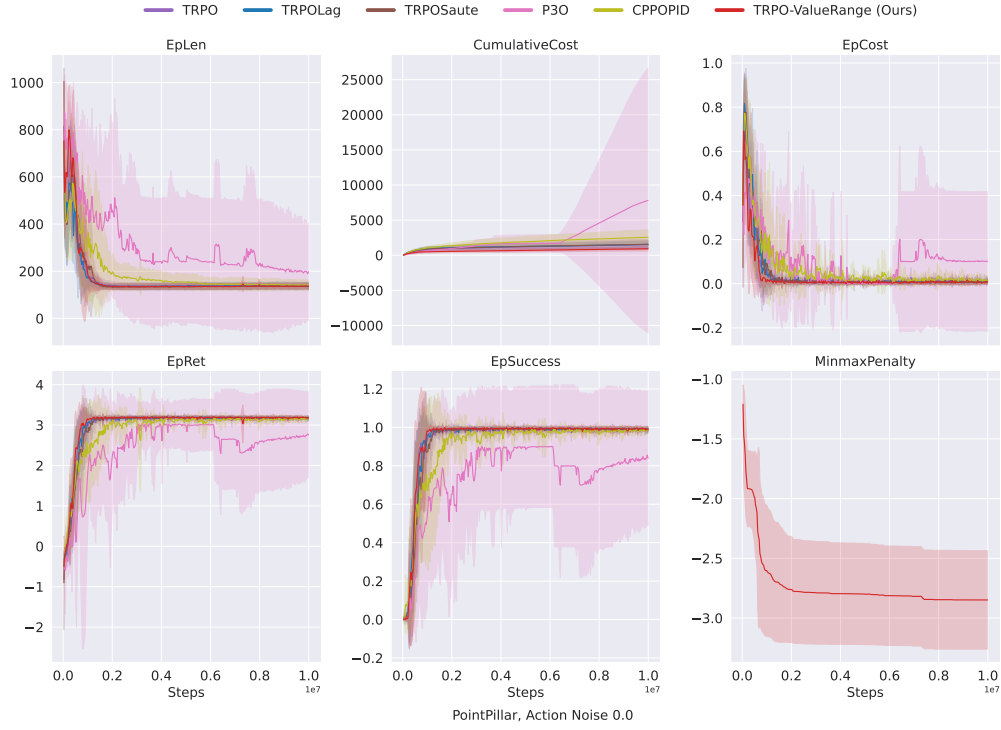
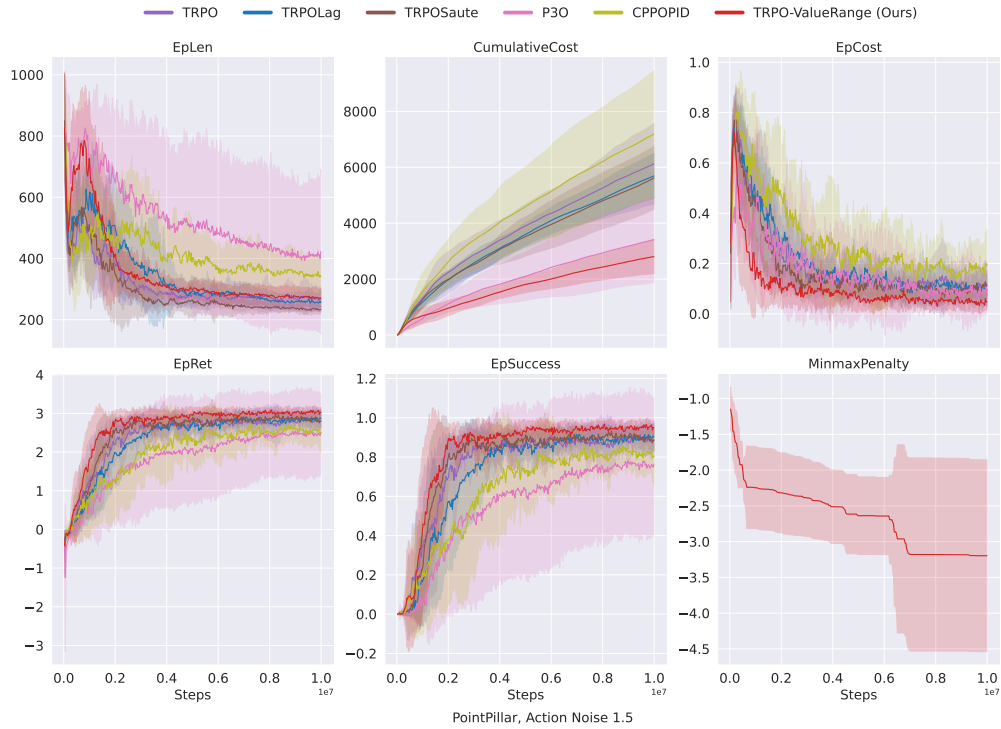
8 Training curves and PPO results with OmniSafe (Ji et al., 2024) baselines

Task	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps
PG	Ours (TRPO)	0.04 ± 0.03	0.50 ± 0.17	0.84 ± 0.45	532.08 ± 148.18
	Ours (PPO)	0.04 ± 0.02	0.84 ± 0.06	1.64 ± 0.18	253.69 ± 49.72
	TRPO-Lagrangian	0.09 ± 0.03	0.86 ± 0.03	1.76 ± 0.08	119.18 ± 19.82
	Sauté-TRPO	0.12 ± 0.03	0.87 ± 0.03	1.77 ± 0.09	77.97 ± 10.33
	TRPO	0.10 ± 0.02	0.90 ± 0.02	1.84 ± 0.04	73.86 ± 4.37
	CPPOPID	0.11 ± 0.03	0.89 ± 0.03	1.81 ± 0.08	73.65 ± 5.04
	P3O	0.08 ± 0.02	0.91 ± 0.02	1.86 ± 0.06	101.98 ± 13.03
PP	Ours (TRPO)	0.08 ± 0.03	0.01 ± 0.01	0.47 ± 0.11	940.58 ± 20.33
	Ours (PPO)	0.12 ± 0.07	0.09 ± 0.14	1.12 ± 1.30	927.77 ± 31.18
	TRPO-Lagrangian	0.12 ± 0.05	0.03 ± 0.03	0.62 ± 0.21	914.53 ± 29.49
	Sauté-TRPO	0.13 ± 0.05	0.08 ± 0.14	0.92 ± 0.73	905.51 ± 33.87
	TRPO	0.14 ± 0.06	0.05 ± 0.06	0.72 ± 0.37	903.23 ± 37.82
	CPPOPID	0.13 ± 0.04	0.04 ± 0.03	0.52 ± 0.11	889.25 ± 38.75
	P3O	0.13 ± 0.04	0.06 ± 0.05	0.76 ± 0.33	921.17 ± 26.78

Table 5: Results for episodic goal-reaching tasks. Irrespective of the base algorithms used (TRPO-ValueRange or PPO-ValueRange), our approach prioritizes constraint satisfaction, maintaining significantly lower costs than baselines in these complex safety-critical environments.

Task	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps ↑
PG	Ours (TRPO)	0.08 ± 0.05	0.45 ± 0.16	1.87 ± 1.60	950.31 ± 31.45
	Ours (PPO)	0.13 ± 0.05	0.63 ± 0.13	4.45 ± 2.45	927.75 ± 28.89
	TRPO-Lagrangian	0.62 ± 0.07	0.34 ± 0.06	10.17 ± 0.77	607.44 ± 56.15
	Sauté-TRPO	0.79 ± 0.03	0.21 ± 0.03	11.01 ± 0.55	493.01 ± 24.51
	TRPO	0.78 ± 0.05	0.22 ± 0.05	10.68 ± 0.74	483.42 ± 33.07
	CPPOPID	0.80 ± 0.04	0.20 ± 0.03	10.31 ± 0.88	482.05 ± 36.60
	P3O	0.56 ± 0.07	0.42 ± 0.07	10.63 ± 0.74	667.08 ± 49.14
PP	Ours (TRPO)	0.09 ± 0.03	0.05 ± 0.04	0.53 ± 0.15	905.26 ± 20.12
	Ours (PPO)	0.10 ± 0.02	0.03 ± 0.02	0.52 ± 0.07	914.64 ± 16.92
	TRPO-Lagrangian	0.11 ± 0.03	0.13 ± 0.18	0.83 ± 0.49	844.21 ± 110.39
	Sauté-TRPO	0.12 ± 0.05	0.10 ± 0.12	0.68 ± 0.30	838.98 ± 106.42
	TRPO	0.15 ± 0.07	0.16 ± 0.21	0.86 ± 0.56	795.70 ± 157.18
	CPPOPID	0.16 ± 0.03	0.04 ± 0.01	0.54 ± 0.09	896.50 ± 19.31
	P3O	0.13 ± 0.06	0.11 ± 0.12	0.78 ± 0.36	859.75 ± 74.76

Table 6: Performance in continual goal tasks. Irrespective of the base algorithms used (TRPO-ValueRange or PPO-ValueRange), our approach demonstrates superior survival rates (Total Steps) and consistent safety regulation compared to existing constrained RL methods.

Figure 5: Training curves using OmniSafe in the PILLAR environment with $\text{noise} = 0$ Figure 6: Training curves using OmniSafe in the PILLAR environment with $\text{noise} = 1.5$

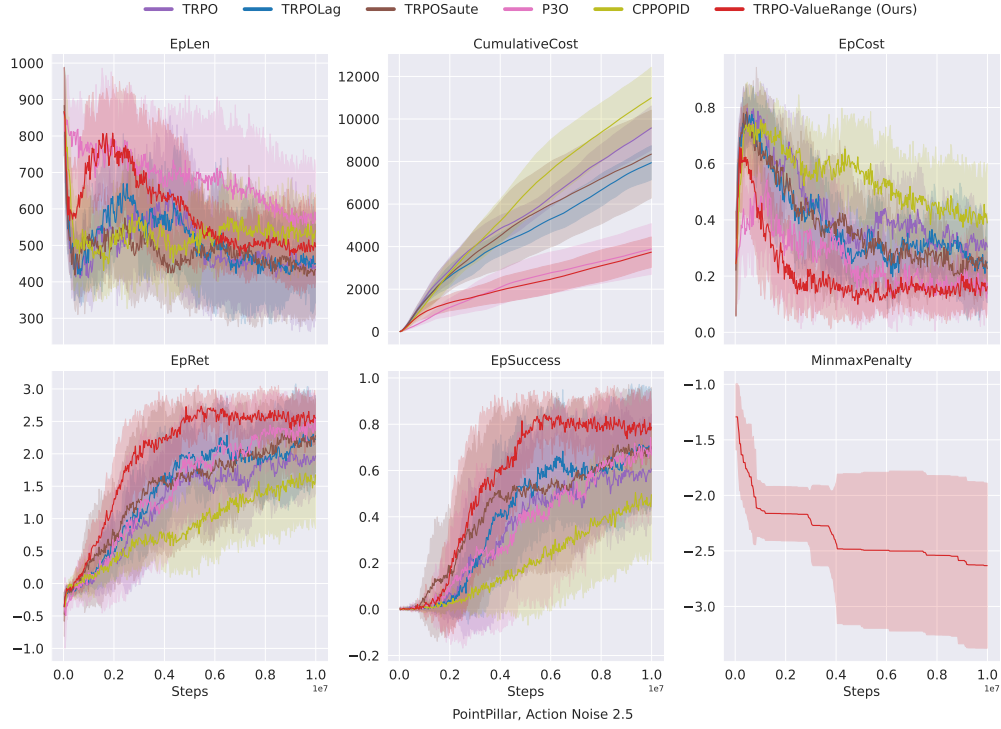


Figure 7: Training curves using OmniSafe in the PILLAR environment with $noise = 2.5$

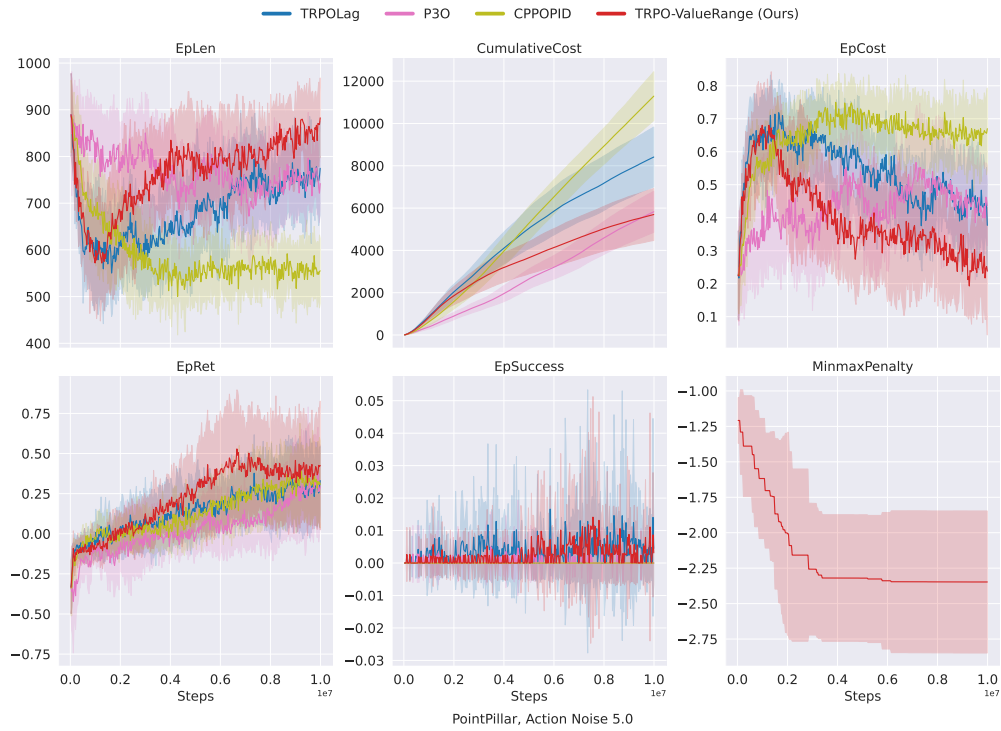
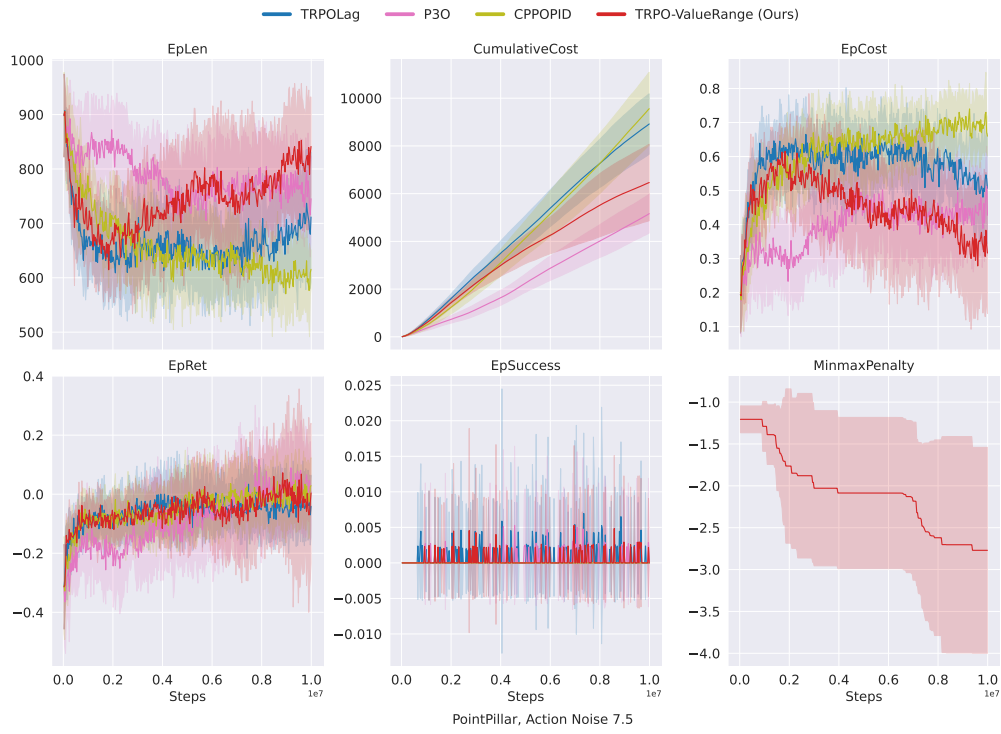
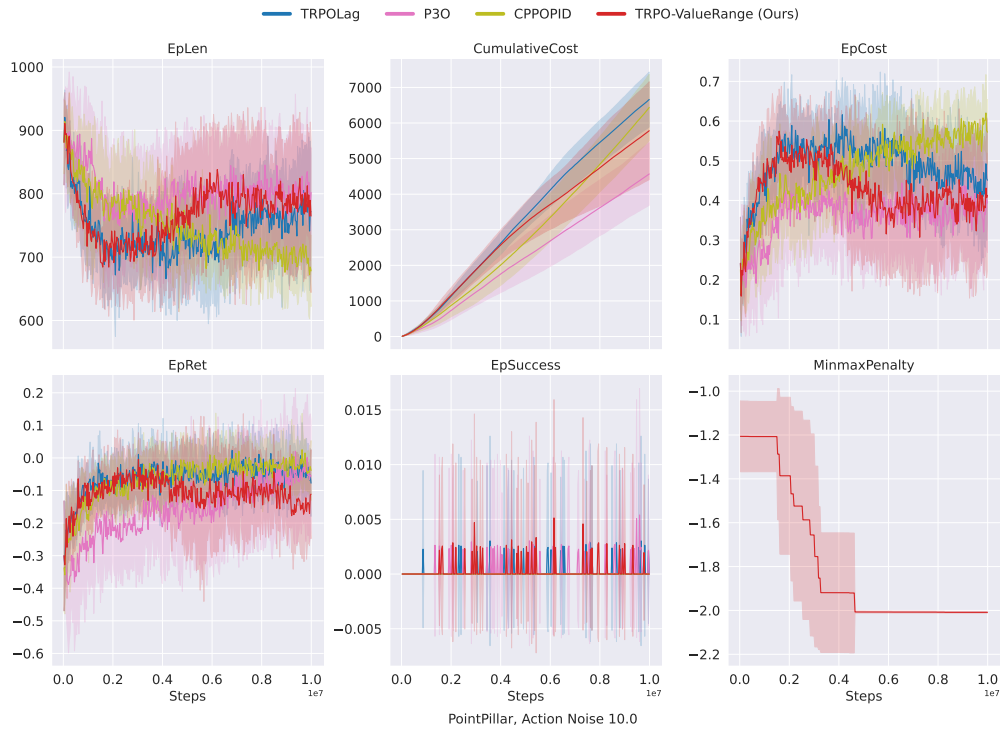


Figure 8: Training curves using OmniSafe in the PILLAR environment with $noise = 5$

Figure 9: Training curves using OmniSafe in the PILLAR environment with $\text{noise} = 7.5$ Figure 10: Training curves using OmniSafe in the PILLAR environment with $\text{noise} = 10$

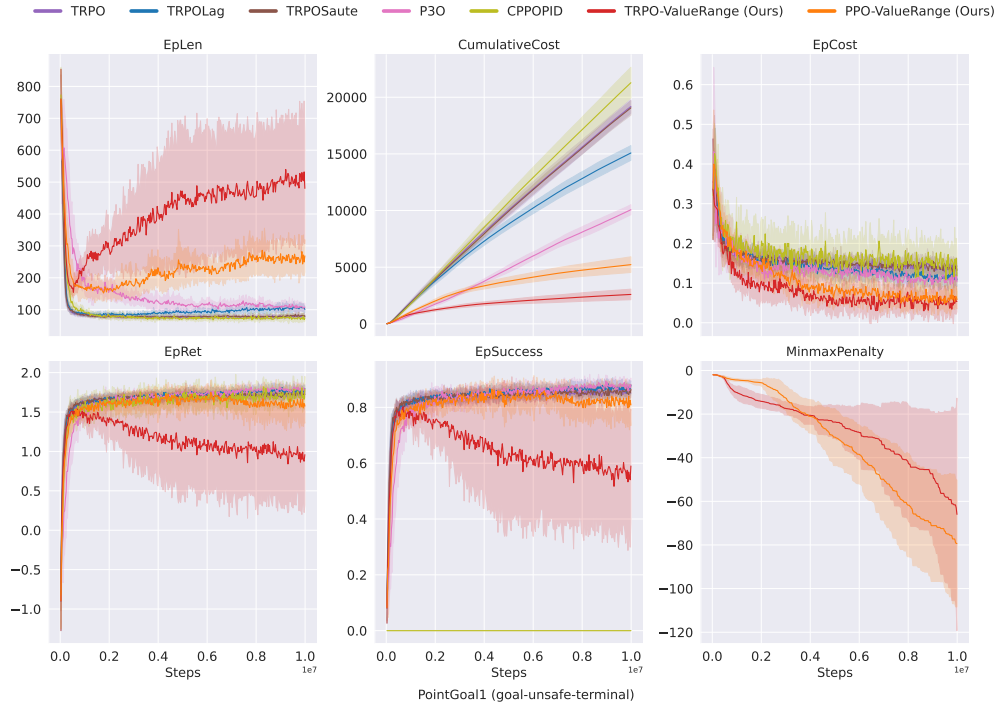


Figure 11: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTGOAL1 environment, modified to terminate when the goal is reached and in unsafe transitions.

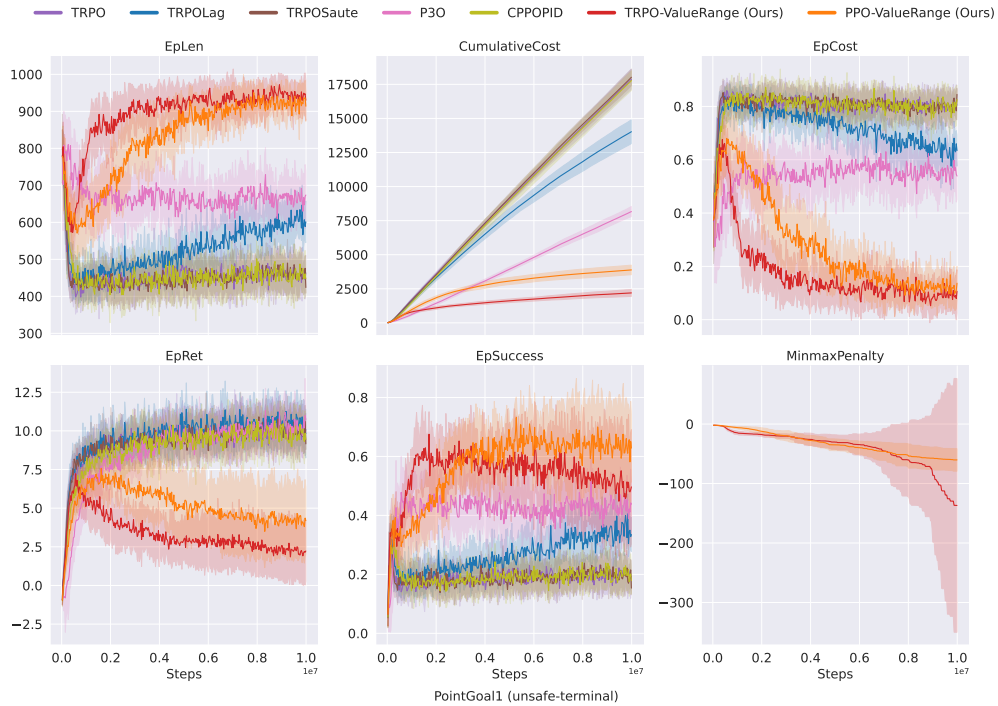


Figure 12: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTGOAL1 environment, modified to terminate in unsafe transitions.

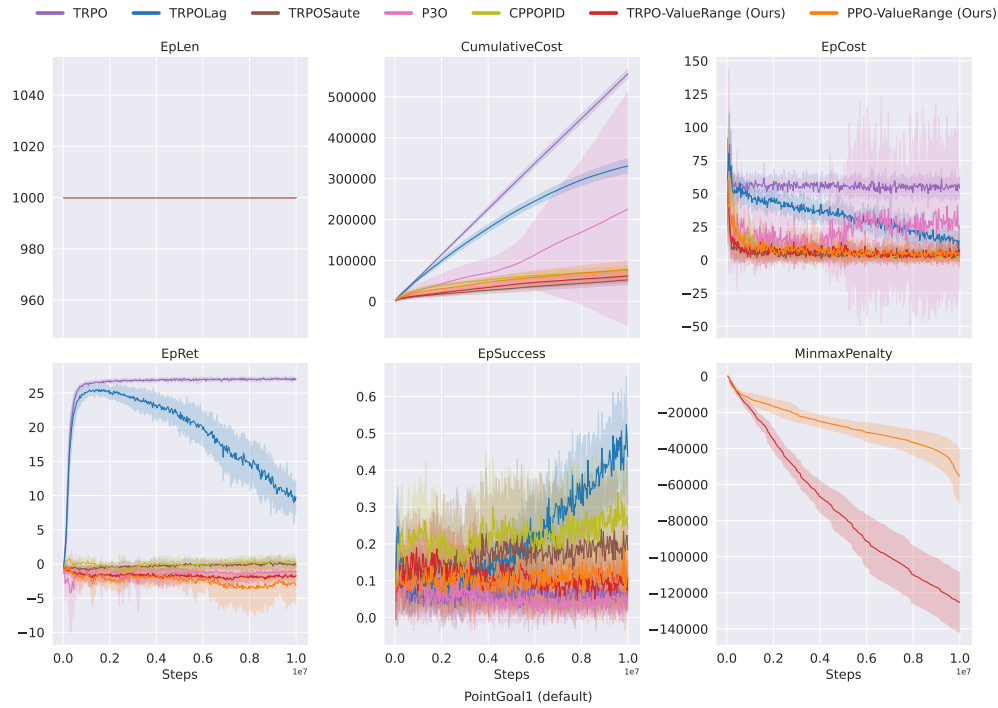


Figure 13: Training curves for trained models with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTGOAL1 environment, with no termination.

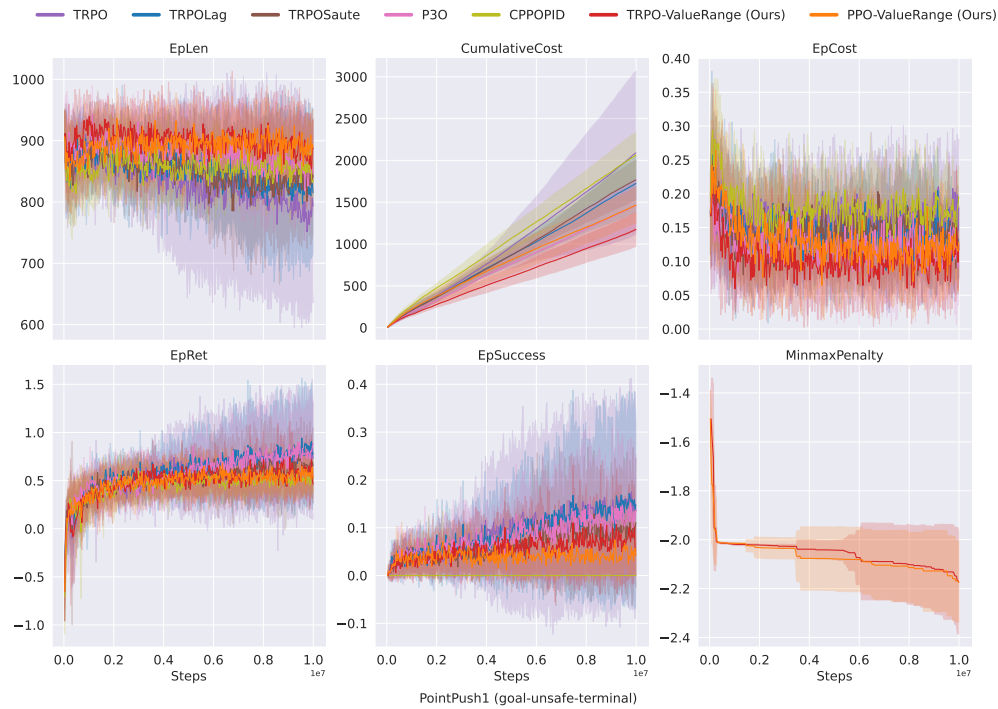


Figure 14: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTPUSH1 environment, modified to terminate when the goal is reached and in unsafe transitions.

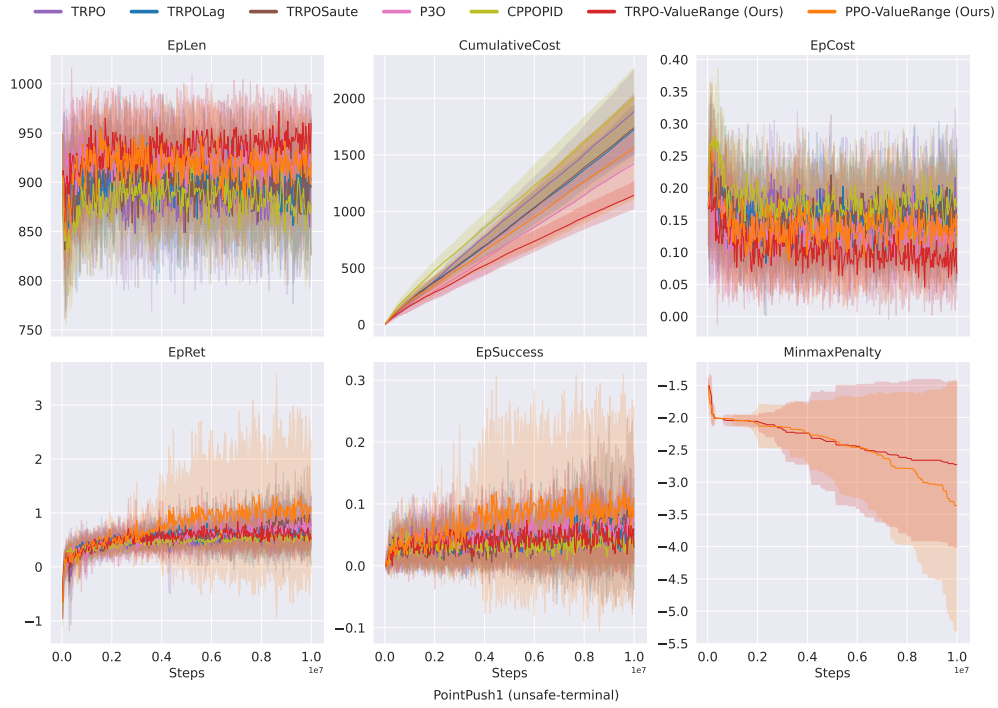


Figure 15: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTPUSH1 environment, modified to terminate in unsafe transitions.

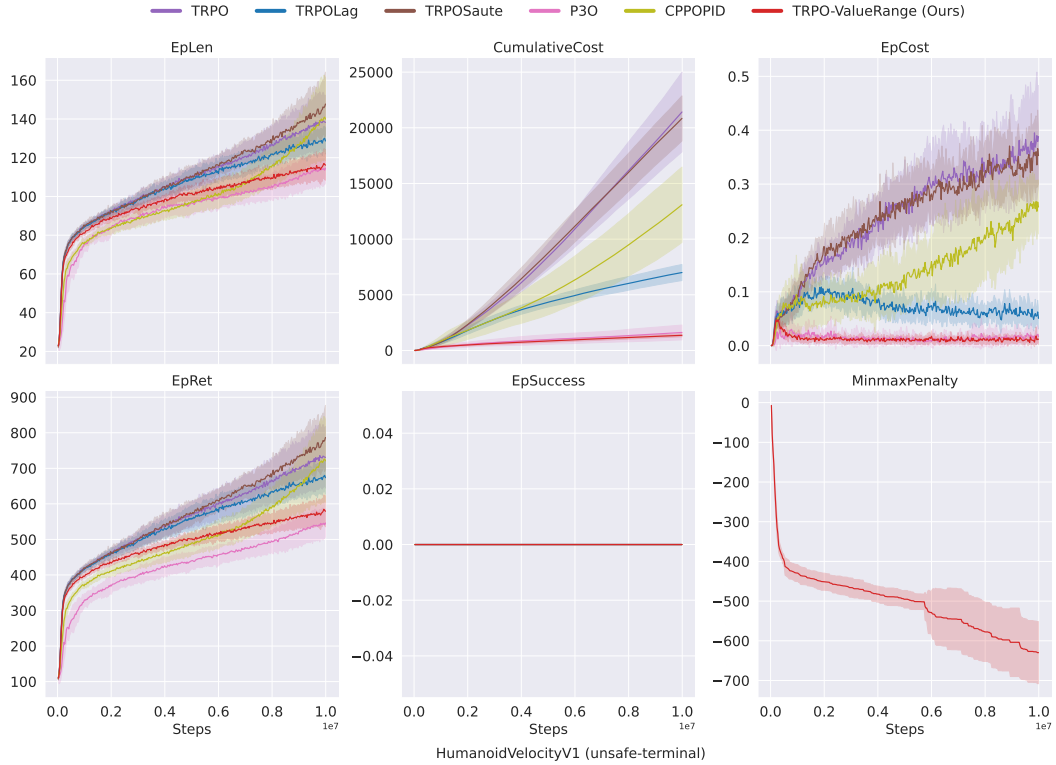


Figure 16: Training curves using OmniSafe in the HUMANOIDVELOCITY environment

9 Reproducibility with other baseline implementations

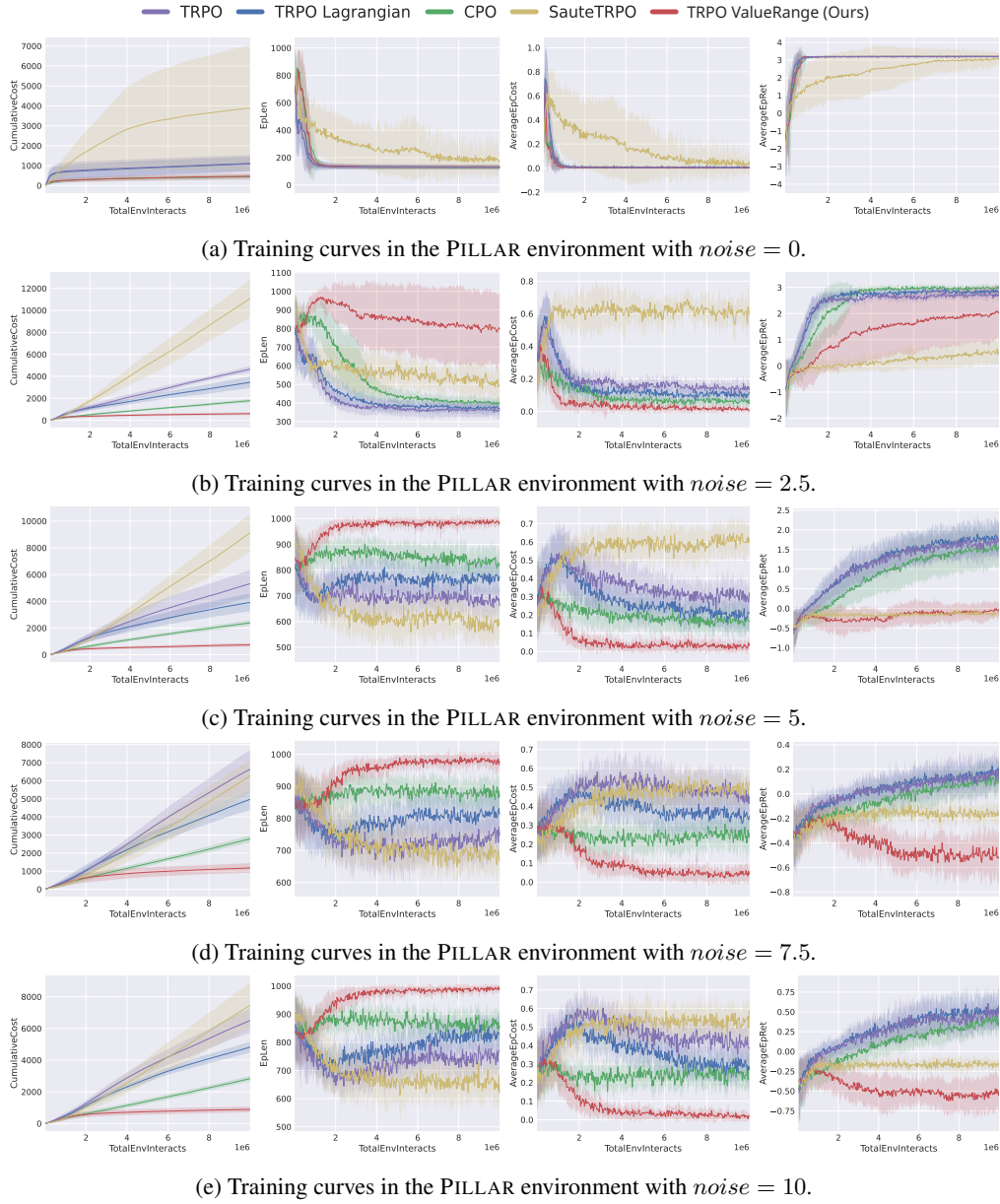


Figure 17: Training curves with Ray et al. (2019) baselines in the PILLAR environment. We observe that in the deterministic case $noise = 0$, all the algorithms achieve similar performance (except Sauté-TRPO), successfully maximising returns while minimising the failure rates. However, for the stochastic cases $noise > 0$, we can observe that all the baselines except Sauté-TRPO achieve significantly high returns at the expense of a rapidly increasing cumulative cost. Finally, by examining the episode length and failure rates for all the baselines in the stochastic cases, we can conclude that they have all learned risky policies that maximise rewards over short trajectories that are highly likely to result in collisions. In contrast, the results obtained show that TRPO-ValueRange successfully solves the tasks while minimising cost for both deterministic and stochastic environments, when the noise levels are not too high ($noise \in [0, 2.5]$). In addition, we can observe from the episode lengths that TRPO-ValueRange chooses the shortest path to the goal when there is no noise, but chooses longer paths as the noise increases.

Noise	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps
0.0	TRPO	0.00 ± 0.00	1.00 ± 0.00	3.21 ± 0.00	130.30 ± 14.94
	TRPO-Lagrangian	0.00 ± 0.01	1.00 ± 0.01	3.20 ± 0.02	132.16 ± 14.43
	CPO	0.00 ± 0.00	1.00 ± 0.00	3.21 ± 0.01	128.06 ± 14.40
	Sauté-TRPO	0.04 ± 0.19	0.95 ± 0.21	3.09 ± 0.55	176.51 ± 117.93
	TRPO-ValueRange	0.00 ± 0.00	1.00 ± 0.00	3.21 ± 0.01	131.53 ± 15.15
2.5	TRPO	0.18 ± 0.03	0.82 ± 0.03	2.58 ± 0.12	351.33 ± 40.17
	TRPO-Lagrangian	0.13 ± 0.03	0.86 ± 0.02	2.73 ± 0.09	364.41 ± 32.24
	CPO	0.08 ± 0.03	0.92 ± 0.03	2.91 ± 0.10	393.36 ± 29.50
	Sauté-TRPO	0.62 ± 0.49	0.16 ± 0.37	0.59 ± 1.27	484.24 ± 340.57
	TRPO-ValueRange	0.02 ± 0.02	0.47 ± 0.38	2.00 ± 1.02	799.41 ± 181.46
5.0	TRPO	0.32 ± 0.07	0.41 ± 0.16	1.66 ± 0.43	665.62 ± 38.34
	TRPO-Lagrangian	0.20 ± 0.07	0.39 ± 0.16	1.78 ± 0.47	760.66 ± 43.54
	CPO	0.18 ± 0.04	0.27 ± 0.21	1.53 ± 0.54	807.28 ± 51.38
	Sauté-TRPO	0.62 ± 0.49	0.01 ± 0.07	-0.09 ± 0.54	594.09 ± 363.81
	TRPO-ValueRange	0.05 ± 0.03	0.00 ± 0.00	-0.00 ± 0.19	975.59 ± 17.81
7.5	TRPO	0.43 ± 0.06	0.02 ± 0.03	0.45 ± 0.21	726.97 ± 31.42
	TRPO-Lagrangian	0.30 ± 0.06	0.01 ± 0.01	0.55 ± 0.18	806.91 ± 41.44
	CPO	0.28 ± 0.04	0.00 ± 0.01	0.38 ± 0.13	830.78 ± 25.03
	Sauté-TRPO	0.54 ± 0.50	0.00 ± 0.03	-0.15 ± 0.48	650.94 ± 364.90
	TRPO-ValueRange	0.02 ± 0.02	0.00 ± 0.00	-0.46 ± 0.20	989.69 ± 7.78
10.0	TRPO	0.46 ± 0.08	0.00 ± 0.00	0.13 ± 0.11	725.03 ± 49.64
	TRPO-Lagrangian	0.36 ± 0.09	0.00 ± 0.00	0.17 ± 0.09	789.52 ± 42.68
	CPO	0.27 ± 0.06	0.00 ± 0.00	0.10 ± 0.10	859.58 ± 30.94
	Sauté-TRPO	0.46 ± 0.50	0.00 ± 0.00	-0.18 ± 0.48	701.60 ± 355.32
	TRPO-ValueRange	0.07 ± 0.05	0.00 ± 0.00	-0.48 ± 0.20	960.96 ± 28.39

Table 7: Evaluation of trained models with [Ray et al. \(2019\)](#) baselines in the PILLAR environment with varying noise levels. Figure 17 shows the training curves. Here, higher episode lengths are better for $noise > 0$ because that means the policy is taking longer safer paths. We observe that only TRPO-ValueRange prioritises minimising the probability of unsafe transitions, consistently achieving the lowest cost while trading off the rewards. It achieves the same highest success rate as the baselines only in the deterministic case, since the pure maximisation of rewards here doesn’t come at the cost of higher unsafe transitions. It also does not completely ignore the rewards when the noise is not too large ($noise = 2.5$). We can also observe from the training curves of $noise = 2.5$ (Figure 17b) that TRPO-ValueRange has not converged in its rewards performance and is still increasing.

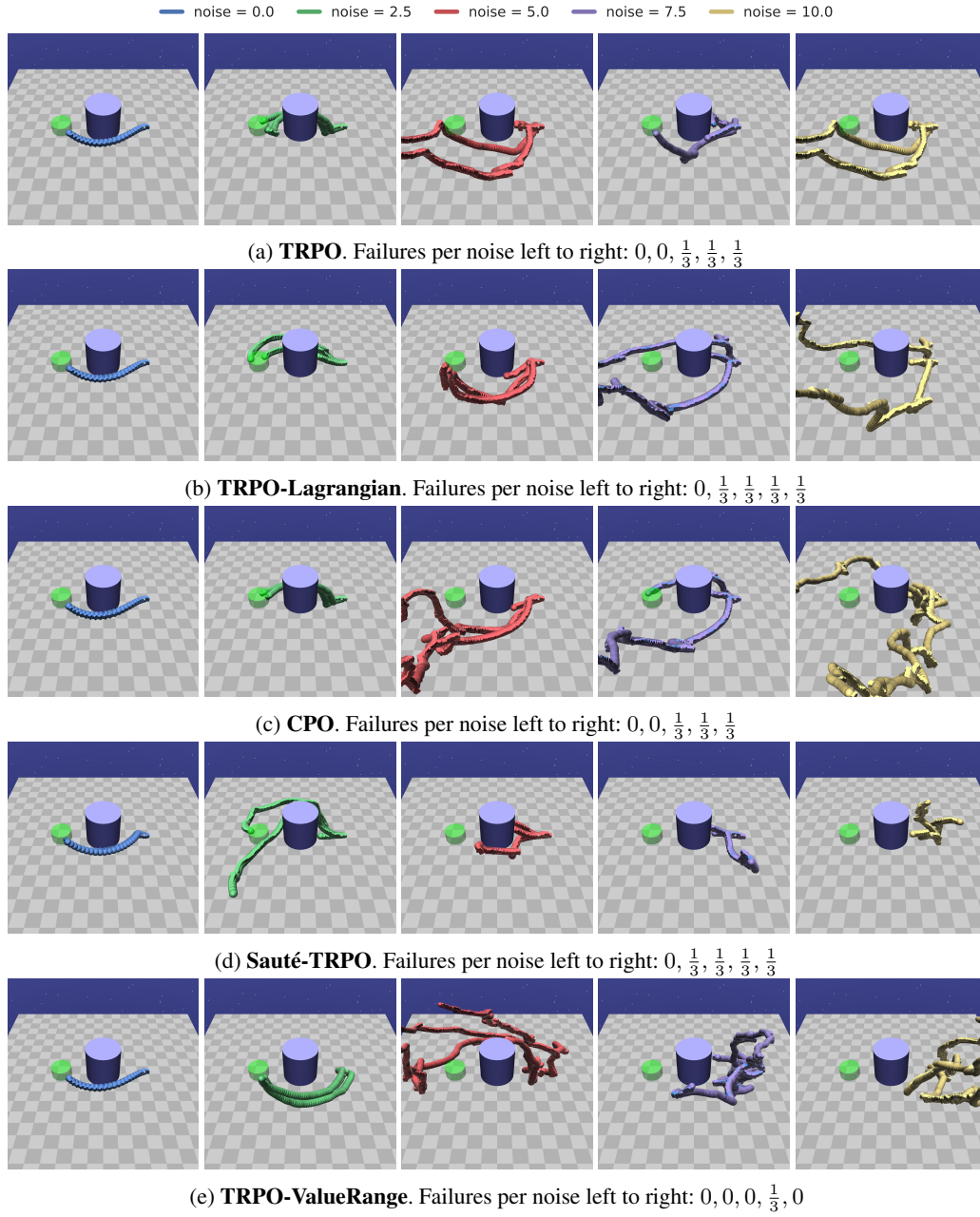


Figure 18: Sample trajectories of policies learned by each baseline and our **TRPO-ValueRange** approach in the Safety Gym PILLAR environment with varying noise levels. To sample the trajectories for each noise level, we use the same three environment random seeds across all the algorithms. We can observe that $noise \geq 5$ is too noisy to learn safe policies, at least after 10 million training steps.

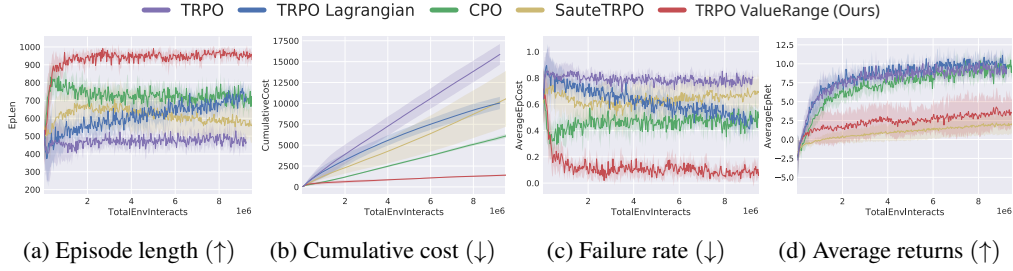


Figure 19: Comparison with Ray et al. (2019) baselines in POINTGOAL1. Here, higher episode lengths are better because episodes only terminate when the agent reaches $\mathcal{G}^1$ or after 1000 timesteps. Similar to Figure 17, all the baselines except Sauté-RL achieve significantly high returns at the expense of a rapidly increasing cumulative cost. By comparison, TRPO-ValueRange dramatically reduces the failure rate while still being able to solve the task, as observed by average returns achieved as well as the trajectories observed. However, returns are lower since TRPO-ValueRange learns safer longer paths to the goals (see sample trajectories in Figure 23).

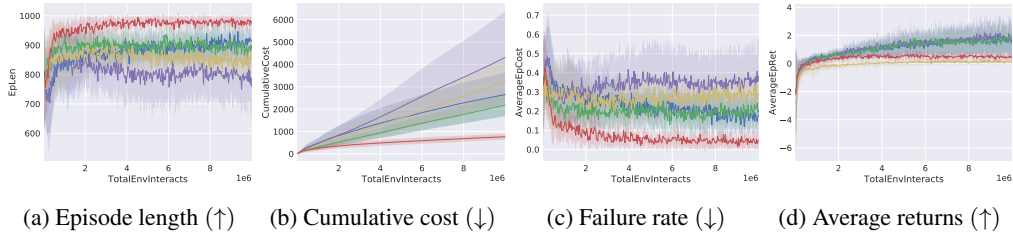


Figure 20: Comparison with Ray et al. (2019) baselines in POINTPUSH1. Similar to Figure 17, the baselines achieve higher returns at the expense of a rapidly increasing cumulative cost while TRPO-ValueRange consistently prioritises maintaining low failure rates by sacrificing rewards.

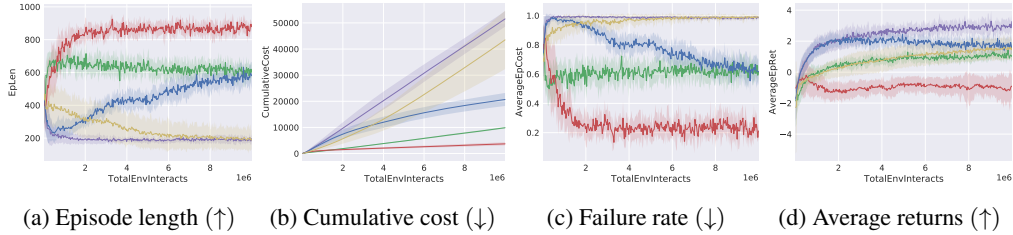


Figure 21: Comparison with Ray et al. (2019) baselines in POINTBUTTON1. Similar to Figure 17, the baselines achieve significantly high returns at the expense of a rapidly increasing cumulative cost while TRPO-ValueRange consistently prioritises maintaining low failure rates.

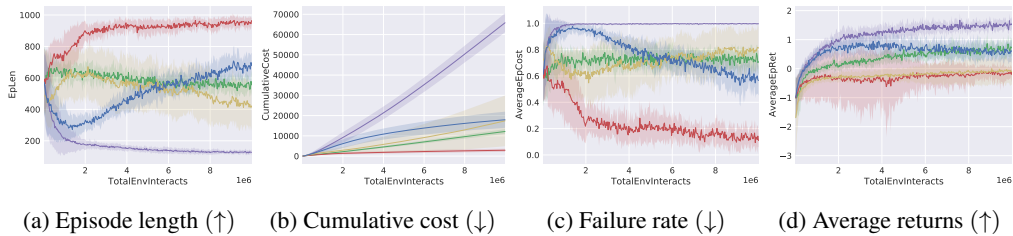


Figure 22: Comparison with Ray et al. (2019) baselines in CARBUTTON1. Similar to Figure 17, the baselines achieve significantly high returns at the expense of a rapidly increasing cumulative cost while TRPO-ValueRange consistently prioritises maintaining low failure rates.

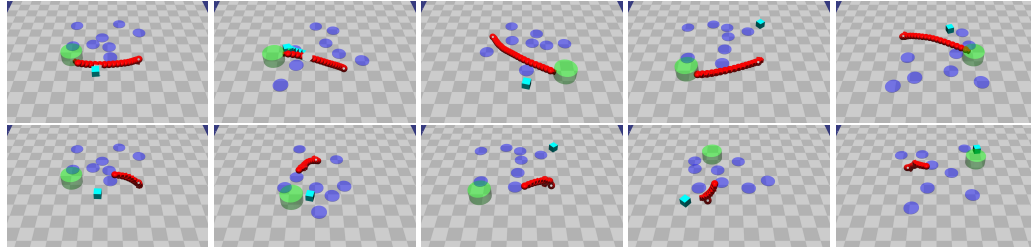
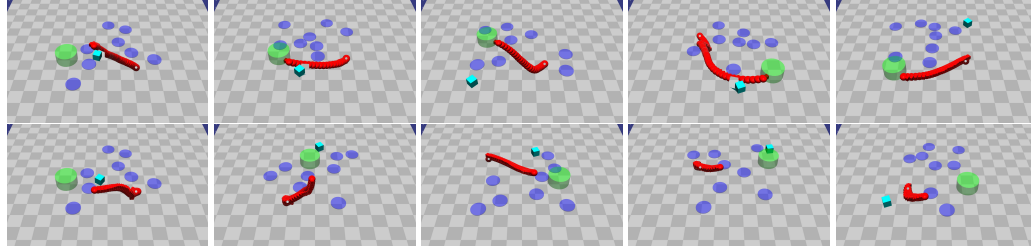
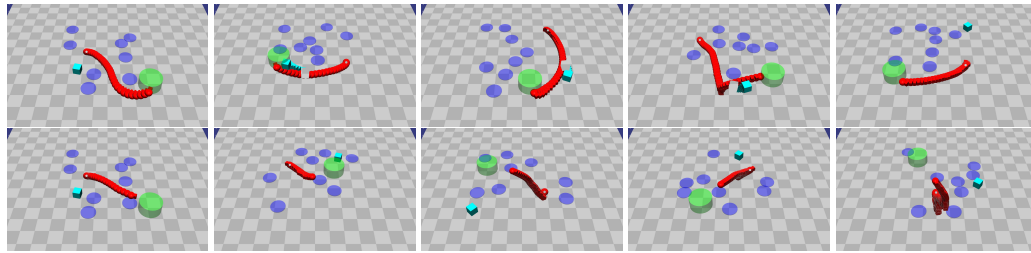
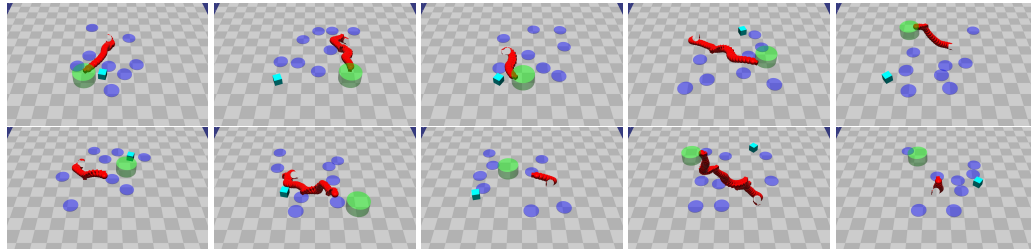
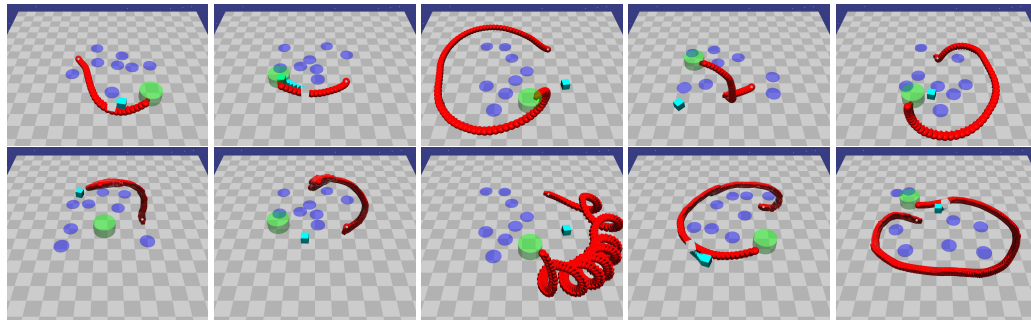
(a) **TRPO** successes (top) and failures (bottom)(b) **TRPO-Lagrangian** successes (top) and failures (bottom)(c) **CPO** successes (top) and failures (bottom)(d) **Sauté-TRPO** successes (top) and failures (bottom)(e) **TRPO-ValueRange** successes (top) and failures (bottom)

Figure 23: Sample trajectories of policies learned by each baseline and our ValueRange approach in the Safety Gym POINTGOAL1 domain, in the experiments of Figure 19. Trajectories that hit hazards or take more than 1000 timesteps to reach the goal location are considered failures.

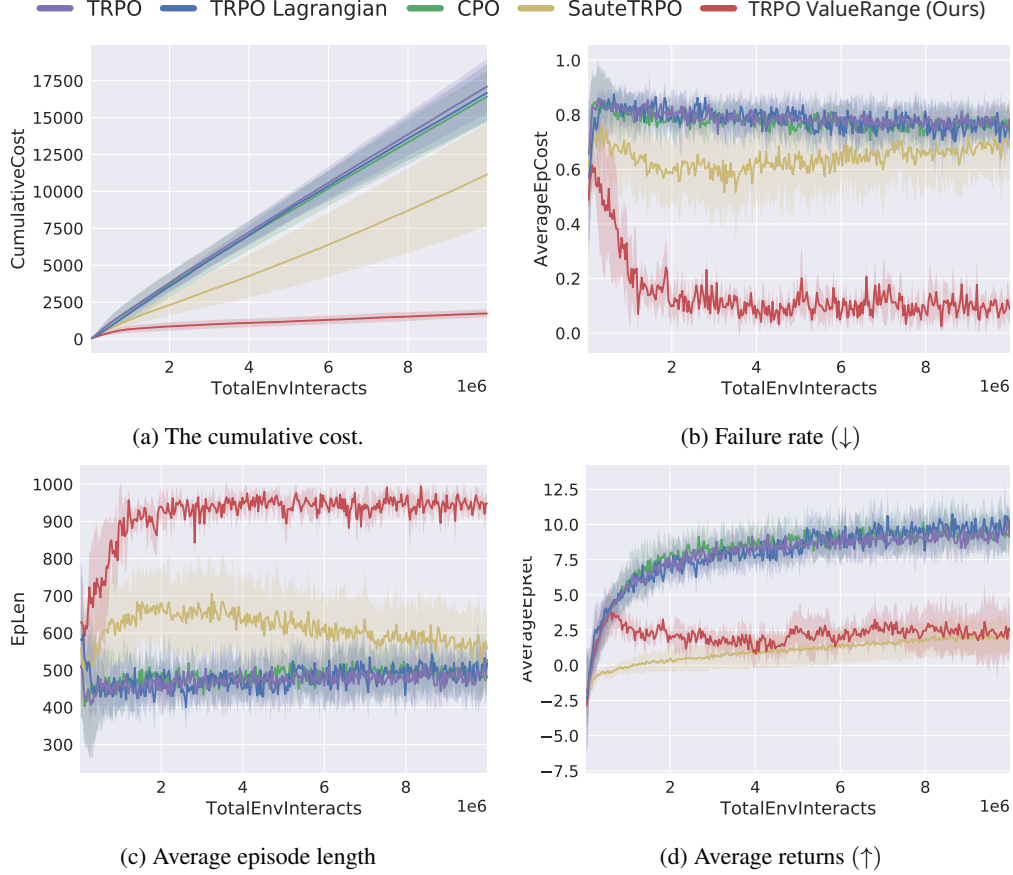


Figure 24: Comparison with Ray et al. (2019) baselines in POINTGOAL1. Here, higher episode lengths are better since episodes only terminate when the agent reaches a hazard or after 1000 timesteps. This experiment is similar to Figure 19, but instead of a cost threshold of 0, it uses a cost threshold of 25 for the baselines (as in Ray et al. (2019)) to check its effect on the performance of the baselines when episodes immediately terminate at unsafe states. We can observe drastically worse failure rates and cumulative costs for the baselines compared to their performance in Figure 19. Similar results were obtained when using a cost threshold of 1. These show how sensitive such approaches are to the cost threshold, while a reward only approach like TRPO-ValueRange does not depend on such hyperparameters.

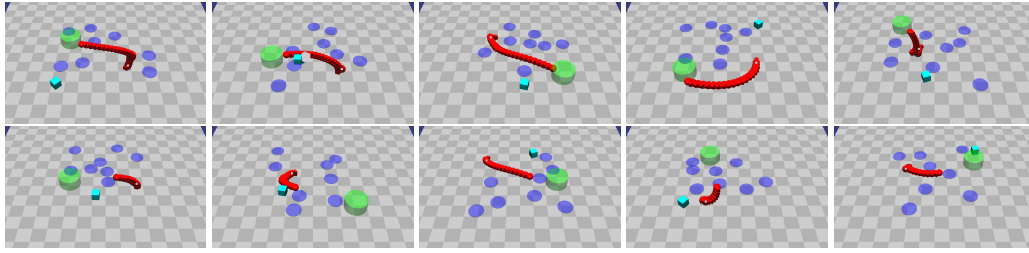
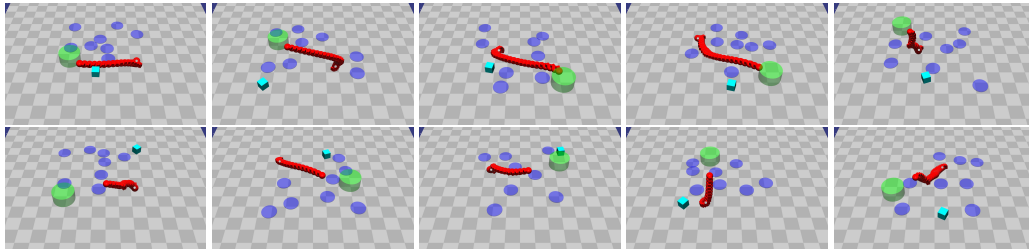
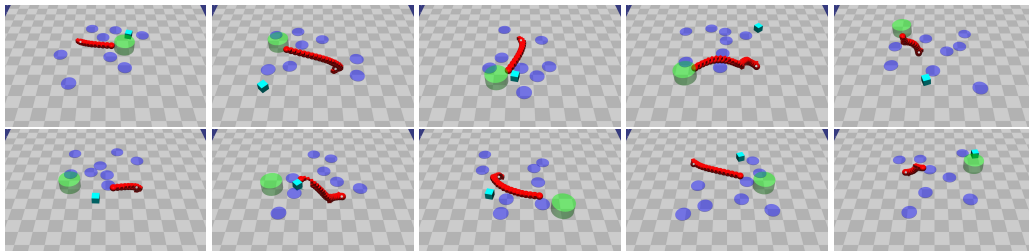
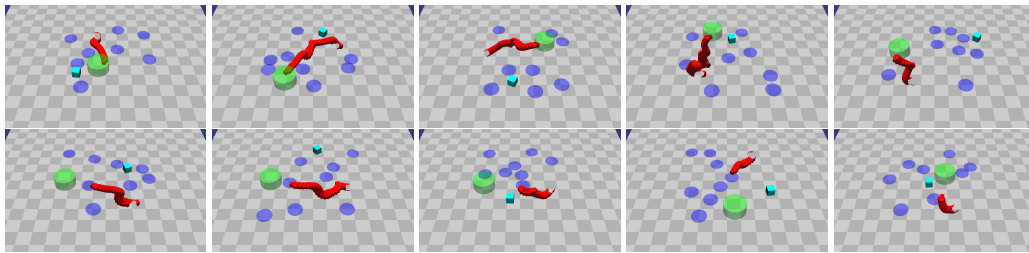
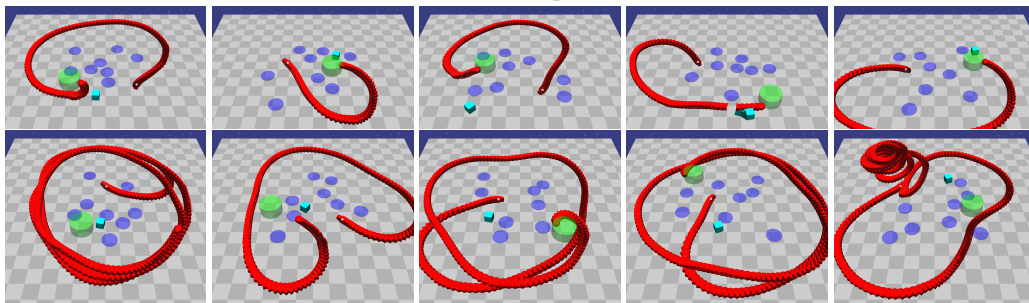
(a) **TRPO** successes (top) and failures (bottom)(b) **TRPO-Lagrangian** successes (top) and failures (bottom)(c) **CPO** successes (top) and failures (bottom)(d) **Sauté-TRPO** successes (top) and failures (bottom)(e) **TRPO-ValueRange** successes (top) and failures (bottom)

Figure 25: Sample trajectories of policies learned by each baseline and our ValueRange approach in the Safety Gym POINTGOAL1 domain, in the experiments of Figure 24. Trajectories that hit hazards or take more than 1000 timesteps to reach the goal location are considered failures.