Efficient Risk-Averse Reinforcement Learning

Ido Greenberg
Technion
gido@campus.technion.ac.il

Yinlam Chow
Google Research
yinlamchow@google.com

Mohammad Ghavamzadeh
Google Research
ghavamza@google.com

Shie Mannor
Technion, Nvidia Research
shie@ee.technion.ac.il

Abstract

In risk-averse reinforcement learning (RL), the goal is to optimize some risk measure of the returns. A risk measure often focuses on the worst returns out of the agent's experience. As a result, standard methods for risk-averse RL often ignore high-return strategies. We prove that under certain conditions this inevitably leads to a local-optimum barrier, and propose a mechanism we call soft risk to bypass it. We also devise a novel cross entropy module for sampling, which (1) preserves risk aversion despite the soft risk; (2) independently improves sample efficiency. By separating the risk aversion of the sampler and the optimizer, we can *sample* episodes with poor conditions, yet *optimize* with respect to successful strategies. We combine these two concepts in CeSoR – Cross-entropy Soft-Risk optimization algorithm – which can be applied on top of *any* risk-averse policy gradient (PG) method. We demonstrate improved risk aversion in maze navigation, autonomous driving, and resource allocation benchmarks, including in scenarios where standard risk-averse PG completely fails. Our results and CeSoR implementation are available on Github. The stand-alone cross entropy module is available on PyPI.

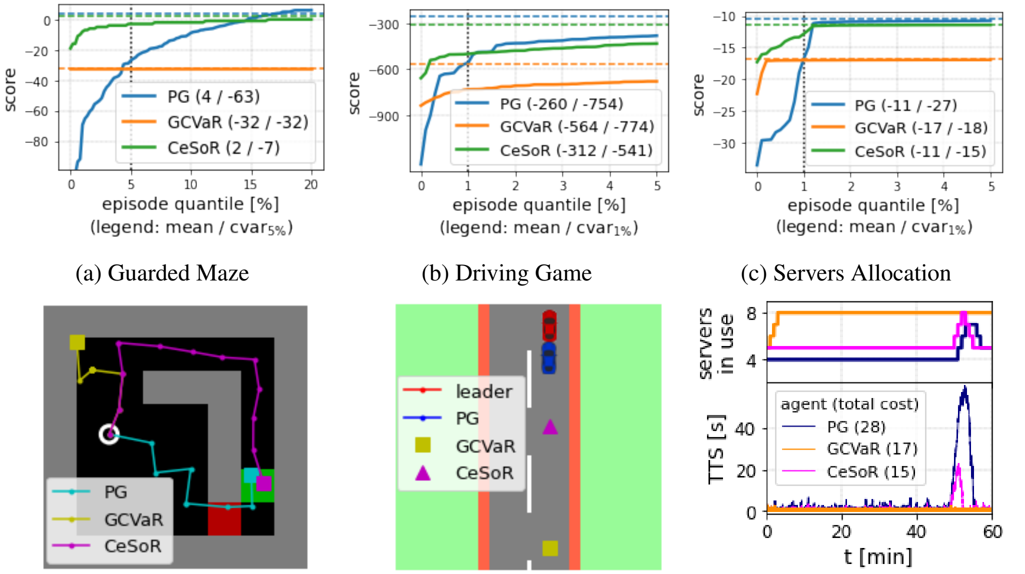
1 Introduction

Risk-averse reinforcement learning (RL) is important for high-stake applications, such as driving, robotic surgery, and finance [Vittori et al., 2020]. In contrast to risk-neutral RL, it optimizes a risk measure of the return random variable, rather than its expectation. A popular risk measure is the Conditional Value at Risk (CVaR), defined as $\mathrm{CVaR}_\alpha(R) = \mathbb{E}\left[R \,|\, R \leq q_\alpha(R)\right]$, where $q_\alpha(R) = \inf\{x \,|\, F_R(x) \geq \alpha\}$ is the α-quantile of the random variable R and F_R is its CDF. Intuitively, CVaR measures the expected return below a specific quantile α, also termed the risk level. CVaR optimization is widely researched in the RL community, e.g., using adjusted policy gradient approaches (CVaR-PG) [Tamar et al., 2015b, Hiraoka et al., 2019]. In addition, CVaR is a coherent risk measure, and its optimization is equivalent to a robust optimization problem [Chow et al., 2015].

Since risk-averse RL aims to avoid the hazardous parts of the environment (e.g., dangerous areas in navigation), CVaR-PG algorithms typically sample a batch of N trajectories (episodes), and then optimize w.r.t. the mean of the αN trajectories with worst returns [Tamar et al., 2015b, Rajeswaran et al., 2017]. This approach suffers from two major drawbacks: (i) $1-\alpha$ of the batch is wasted and excluded from the optimization (where often $0.01 \leq \alpha \leq 0.05$), leading to sample inefficiency; (ii) focusing on the worst episodes inherently overlooks good agent strategies corresponding to high returns – a phenomenon we refer to as the *blindness to success*.

An illustrative example – the Guarded Maze: Consider the Guarded Maze benchmark (visualized in Figure 1d). The goal is to reach the target zone (a constant location marked in green), resulting in

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) Guarded Maze (b) Driving Game (c) Servers Allocation

(d) CeSoR learns to avoid the risk (red) and take the long path to the target (green), whereas GCVaR fails due to *blindness to success*.

(e) CeSoR maintains a safe margin from the leader, while PG has an accident and GCVaR maintains a too conservative distance.

(f) CeSoR handles the exceptional peak in user-requests without paying for as many servers as GCVaR, while PG fails to handle the peak.

Figure 1: Over 3 benchmarks, test results of 3 agents: risk-neutral PG, standard CVaR-PG (GCVaR, Tamar et al. [2015b]), and our CeSoR. Top: the lower quantiles of the returns distributions. Bottom: sample episodes.

a reward of 16 points. However, the guarded zone (in red) *may* be watched by a guard who demands a payment from any agent that passes by. Every episode, the probability that a guard is present is $\phi_1 = 20\%$, and the payment is exponentially-distributed with average $\phi_2 = 32$. That is, the cost of crossing the guarded zone in a certain episode is $C = C_1 \cdot C_2$, where $C_1 \sim Ber(\phi_1), C_2 \sim Exp(\phi_2)$ are independent and unknown to the agent. The agent starts at a random point at the lower half, and every time-step, observing its location, it selects one an action: left, right, up or down, with an additive control noise. One point is deducted per step, up to 32 deductions.

In this maze, the shortest path maximizes the average return; yet, the longer path is CVaR-optimal, since sometimes short cuts make long delays [Tolkien, 1954]. However, the standard CVaR-PG optimizer (GCVaR in Figure 1) suffers from blindness to success: in a batch of N random episodes, the worst αN returns (e.g., for $\alpha = 5\%$) usually correspond to either encountering a guard in the short path, or not reaching the goal at all. Hence, the desired long path is never even observed by the CVaR-PG optimizer, and cannot be learned.

Our key insight is that the variation in returns comes from both environment conditions (*epistemic* uncertainty) and agent actions (*aleatoric* uncertainty). We wish to focus on the *low quantiles w.r.t. the conditions* (e.g., a costly guard in the short path of the maze), yet to be exposed to the *high quantiles w.r.t. the strategies* (e.g., taking the long path in the maze). To that end, we devise two mechanisms: first, we use a soft risk-level scheduling method, which begins the training with risk neutrality $\alpha' = 1$, and gradually shifts the risk aversion to $\alpha' = \alpha$. Second, we present a novel dynamic-target version of the Cross Entropy method (CE or CEM) [de Boer et al., 2005], aiming to sample the worst parts of the environment. That is, the CEM samples trajectories with more challenging or riskier conditions, and the soft risk feeds a larger part of them ($\alpha' \geq \alpha$) to the CVaR-PG optimizer. Together, these constitute the Cross-entropy method for Soft-Risk optimization (***CeSoR***). CeSoR can be applied on top of any CVaR-PG method to learn any differentiable model (e.g., a neural network).

To apply the CEM, we assume to have certain control over the environment conditions. For example, in driving we may choose the roads for collecting training data, or in any simulation we may control the environment parameters (e.g., ϕ_1, ϕ_2 in the Guarded Maze). Note that (i) only the CE sampler (not the agent) is aware of the conditions; (ii) their underlying effect is unknown to the sampler and may vary with the agent throughout the training, hence the CEM needs to learn it adaptively.

Contribution: We present the following contribution for PG algorithms under risk-sensitive MDP problems (as defined in Section 2):

1. We analyze the phenomenon of *blindness to success* in the standard CVaR-PG, and show that it leads to a local-optimum barrier in certain environments (Section 3.1).
2. We analyze the potential increase in sample efficiency – if we could sample directly from the tail of the returns distribution (Section 3.2).
3. We introduce the CeSoR algorithm (Section 4), which modifies any CVaR-PG method with: (i) a soft risk mechanism preventing blindness to success; (ii) a novel dynamic-CE method that over-samples the riskier realizations of the environment, increasing sample efficiency.
4. We demonstrate the effectiveness of CeSoR in 3 risk-sensitive domains (Section 5), where it learns faster and achieves higher returns (both CVaR and mean) than the baseline CVaR-PG.

1.1 Related Work

Optimizing risk in RL is crucial to enforce safety in decision-making [García and Fernández, 2015, Paduraru et al., 2021]. It has been long studied through various risk criteria, e.g., mean-variance [Sato et al., 2001, Prashanth and Ghavamzadeh, 2013, 2016, Xie et al., 2018], entropic risk measure [Borkar and Meyn, 2002, Borkar and Jain, 2014, Fei et al., 2021] and distortion risk measures [Vijayan and Prashanth, 2021]. Tamar et al. [2015a] derived a PG method for general coherent risk measures, given their risk-envelope representation.

The CVaR risk measure specifically was studied using value iteration [Chow et al., 2015] and distributional RL [Dabney et al., 2018a, Tang et al., 2019, Bodnar et al., 2020] (also discussed in Appendix H). CVaR optimization was also shown equivalent to mean optimization under robustness [Chow et al., 2015], motivating robust-RL methods [Pinto et al., 2017, Godbout et al., 2021]. Yet, PG remains the most popular approach for CVaR optimization in RL [Tamar et al., 2015b, Rajeswaran et al., 2017, Hiraoka et al., 2019, Huang et al., 2021b], and can be flexibly applied to a variety of use-cases, e.g., mixed mean-CVaR criteria [Chow and Ghavamzadeh, 2014] and multi-agent problems [Qiu et al., 2021].

Optimizing the CVaR for risk levels $\alpha \ll 1$ poses a significant sample efficiency challenge, as only a small portion of the agent's experience is used to optimize its policy [Curi et al., 2020]. Keramati et al. [2020] used an exploration-based approach to address the sample efficiency. Pessimistic sampling for improved sample efficiency was suggested heuristically by Tamar et al. [2015b] using a dedicated value function, but no systematic method was suggested to direct the pessimism level. In this work, we use the CEM to control the sampled episodes around the desired risk level α, and demonstrate CVaR optimization for as extreme levels as $\alpha = 1\%$. Note that unlike other CE-optimizers in RL [Mannor et al., 2003, Huang et al., 2021c], we use the CEM for *sampling*, to support a gradient-based optimizer.

2 Problem Formulation

Consider a Markov Decision Process (MDP) (S, A, P, γ, P_0), corresponding to states, actions, state-transition and reward distribution, discount factor, and initial state distribution, respectively. For any policy parameter $\theta \in \mathbb{R}^n$, we denote by π_θ the parameterized policy that maps a state to a probability distribution over actions. Given a state-action-reward trajectory $\tau = \{(s_t, a_t, r_t)\}_{t=0}^{T}$, the trajectory total return is denoted by $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$. The expected return of a policy π_θ is defined as

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim P^{\pi_\theta}} \left[R(\tau) \right], \tag{1}$$

where $P^{\pi_\theta}(\tau) = P_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}, r_t | s_t, a_t) \pi_\theta(a_t | s_t)$ is the probability distribution of τ induced by π_θ. Under the risk-neutral objective, the PG method uses the gradient $\nabla_\theta J(\pi_\theta)$ to learn θ, aiming to increase the probability of actions that lead to higher returns. In contrast, CVaR-PG methods aim to optimize the risk-averse CVaR$_\alpha$ objective (w.r.t. a given risk level α):

$$J_\alpha(\pi_\theta) = \mathbb{E}_{\tau \sim P^{\pi_\theta}} \left[R(\tau) \,|\, R(\tau) \leq q_\alpha(R|\pi_\theta) \right], \tag{2}$$

where $q_\alpha(R|\pi_\theta)$ is the α-quantile of the return random variable of policy π_θ. Thus, CVaR-PG algorithms aim to improve the actions specifically for episodes whose returns are lower than $q_\alpha(R|\pi_\theta)$. Specifically, given a batch of N trajectories $\{\tau_i\}_{i=1}^{N}$ whose empirical return quantile is $\hat{q}_\alpha = \hat{q}_\alpha(\{R(\tau_i)\}_{i=1}^{N})$, the CVaR gradient estimation is given by [Tamar et al., 2015b]:

$$\nabla_\theta \hat{J}_\alpha(\{\tau_i\}_{i=1}^{N}; \pi_\theta) = \frac{1}{\alpha N} \sum_{i=1}^{N} w_i \cdot \mathbf{1}_{R(\tau_i) \leq \hat{q}_\alpha} \left(R(\tau_i) - \hat{q}_\alpha \right) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}; s_{i,t}), \tag{3}$$

where $w_i = P^{\pi_\theta}(\tau_i)/f(\tau_i \,|\, \pi_\theta)$ is the importance sampling (IS) correction factor for τ_i, if τ_i is sampled from a distribution $f \neq P^{\pi_\theta}$. Specifically, as discussed below, we modify the sample distribution using the cross entropy method over a context-MDP formulation of the environment.

Context-MDP: As mentioned above, we aim to focus on high-risk environment conditions. To discuss the notion of conditions, given a standard MDP, we extend its formulation to a Context-MDP (C-MDP) [Hallak et al., 2015], where the *context* is a set of variables that capture (part or all of) the randomness of the original MDP. We define the extension as $(S, A, \mathcal{C}, P_C, \gamma, P_0, D_{\phi_0})$, where $C \in \mathcal{C}$ is sampled from the context space $\mathcal{C}$ according to the distribution D_{ϕ_0} parameterized by ϕ_0, and $P_C(\cdot) = P(\cdot|C)$ is the transition and reward distribution conditioned on C. In a C-MDP, a context-trajectory pair is sampled from the distribution $P^{\pi_\theta}_{\phi_0}(C, \tau) = D_{\phi_0}(C) P^{\pi_\theta}_C(\tau)$, where $P^{\pi_\theta}_C(\tau) = P_0(s_0) \prod_{t=0}^{T-1} P_C(s_{t+1}, r_t | s_t, a_t) \pi_\theta(a_t|s_t)$. The mean and CVaR$_\alpha$ objectives $J(\pi_\theta)$, $J_\alpha(\pi_\theta)$ in Equations (1) and (2) are naturally generalized to C-MDP using the distribution $P^{\pi_\theta}_{\phi_0}(C, \tau)$.

Once we extend an MDP into a C-MDP, we can learn how to modify the context-distribution parameter ϕ to sample high-risk contexts and trajectories, focusing the training on high-risk parts of the environment and thus improving sample efficiency. For this, we assume that certain aspects of the training environment (represented by C) can be controlled. This assumption indeed holds in many practical applications – in both simulated and physical environments. For example, consider a data collection procedure for a self-driving agent training, which by default samples all driving hours uniformly: $C \sim U([0, 24))$. As the hour may affect traffic and driving patterns, a risk-averse driver would prefer to sample more experience in high-risk hours. To that end, we could re-parameterize the uniform distribution as, say, $Beta(\phi)$ with $\phi_0 = (1, 1)$ (note that $Beta(1, 1)$ is the uniform distribution), learn the high-risk hours, and modify ϕ to over-sample them. As another example, in the Guarded Maze described above, we can control the parameters ϕ_1, ϕ_2 of the simulation.

3 Limitations of CVaR-PG

Consider the standard CVaR-PG algorithm, which relies on Equation (3) to apply PG for maximization of $J_\alpha(\pi_\theta)$ of (2). In this section, we analyze two major limitations of this algorithm. Section 3.1 analyzes the *blindness to success* phenomenon, which may bring CVaR-PG learning to a local-optimum deadlock. This will motivate the soft-risk scheduling in Section 4. Section 3.2 analyzes the potential increase in sample efficiency when the environmental context is sampled in correspondence to the tail of the returns distribution. This will motivate the cross-entropy sampler in Section 4.

While the analysis focuses on CVaR-PG methods, Appendix H discusses Distributional RL algorithms for CVaR optimization, and demonstrates that similar limitations apply to these methods as well.

3.1 Blindness to Success

We formally analyze how the *blindness to success* phenomenon can bring the policy learning to a local-optimum deadlock by ignoring successful agent strategies.

Recall the α-quantile of a return distribution $q^\pi_\alpha = \min\{r \mid F_{R(a)|\pi}(r) \geq \alpha\}$. We first introduce the notion of a *tail barrier*, corresponding to a returns-distribution tail with a constant value.

Definition 1 (Tail barrier)**.** Let $\alpha \in (0, 1]$. A policy π has an α-tail barrier if $\forall \alpha' \in [0, \alpha] : q^\pi_{\alpha'} = q^\pi_\alpha$.

Note that in any environment with a discrete rewards distribution, a policy is prone to having a tail barrier for some $\alpha > 0$. In existing CVaR-PG analysis [Tamar et al., 2015b], such barriers are often overlooked by assuming continuous rewards. For the Guarded Maze, Figure 13c in the appendix demonstrates how a standard CVaR-PG exhibits a 0.9-tail barrier, since as many as 90% of the trajectories reach neither the target nor the guard, and thus have identical low returns.

A tail barrier has a destructive effect on CVaR-PG. Consider a CVaR$_\alpha$ objective, and a policy π with a β-tail barrier where $\beta > \alpha$. Intuitively, any infinitesimal change of π cannot affect the CVaR return, since the returns infinitesimally-above q^π_α are identical to those below q^π_α. That is, any tail barrier wider than α brings the CVaR-PG to a stationary point of type plateau. More formally, consider $\nabla_\theta \hat{J}_\alpha$ of Equation (3) with a β-tail barrier $\beta > \alpha$: any trajectory has either $\mathbf{1}_{R(\tau_i) \leq q^\pi_\alpha} = 0$ (if its return is above q^π_α) or $R(\tau_i) - q^\pi_\alpha = 0$ (otherwise), hence the whole gradient vanishes. Such a loss plateau was also observed in a specific MDP in Section 5.1 of Huang et al. [2021a].

In practice, a discrepancy between q_α^π and its estimate $\hat{q}_\alpha(\{R(\tau_i)\})$ (used in Equation 3) may prevent the gradient from completely vanishing, if $q_\alpha^\pi = q_\beta^\pi < \hat{q}_\alpha(\{R(\tau_i)\})$. Otherwise, if $\hat{q}_\alpha(\{R(\tau_i)\}) \le q_\beta^\pi$ in every subsequent iteration, the gradient remains zero, the policy cannot learn any further, and any trajectory returns beyond q_α^π will never be even propagated to the optimizer. We refer to this phenomenon as *blindness to success*.

Definition 2 (Blindness to success)**.** Let a risk level $\alpha \in (0,1)$ and a CVaR-PG training step $m_0 \ge 1$, and let $\beta \in (\alpha, 1)$. Denote by $\mathcal{T}, \Pi$ the spaces of trajectories and policies, respectively, and by $\{\tau_{m,i}\}_{i=1}^N \sim P^{\pi_m}$ the random trajectories in step $m \ge m_0$. We denote by $\mathcal{B}_{\alpha,\beta}^{m_0,n}$ the event of blindness to success in the subsequent n steps (and the complementary event by $\neg\mathcal{B}_{\alpha,\beta}^{m_0,n}$):

$$\mathcal{B}_{\alpha,\beta}^{m_0,n} = \Big\{ \Big\{ \Big(\{\tau_{m,i}\}_{i=1}^N \,,\, \pi_m \Big) \Big\}_{m_0 \le m < m_0+n} \in (\mathcal{T}^N \times \Pi)^n \;\Big|\; \forall m:\; \hat{q}_\alpha(\{R(\tau_{m,i})\}) \le q_\beta^{\pi_{m_0}} \Big\}.$$

Note that Definition 2 uses $q_\beta^{\pi_{m_0}}$ (corresponding to step m_0) to bound the returns in training steps $m > m_0$, thus indeed represents training stagnation. Theorem 1 shows that given a β-tail barrier with $\beta > \alpha$, the probability that CVaR-PG avoids the blindness to success decreases exponentially with $\beta - \alpha$. For example, for $n = 10^6$, $\alpha = 0.05$, $\beta = 0.25$, and $N = 400$, we have $\mathbb{P}(\neg\mathcal{B}_{\alpha,\beta}^{m_0,n}) < 10^{-7}$.

Theorem 1. *Under Definition 2's conditions,* $\mathbb{P}\left(\neg\mathcal{B}_{\alpha,\beta}^{m_0,n} \,\middle|\, \pi_{m_0} \text{ has } \beta\text{-tail barrier}\right) \le ne^{-2N(\beta-\alpha)^2}$.

Proof sketch (see the full proof in Appendix A). In every step m, we have $q_\beta^{\pi_{m_0}} < \hat{q}_\alpha(\{R(\tau_{m,i})\})$ only if at least $1-\alpha$ of the returns are higher than $q_\beta^{\pi_{m_0}}$. We bound the probability of this event using the Hoeffding inequality (Lemma 1). In the complementary event the gradient is 0 (due to the barrier), thus the policy does not change, and the argument can be applied inductively to the next step. □

3.2 Variance Reduction and Sample Efficiency

As discussed in Section 2, an MDP can be often re-parameterized as a C-MDP. In terms of the C-MDP, CVaR-PG samples N context-trajectory pairs from the distribution $P_{\phi_0}^{\pi_\theta}(C,\tau)$, and calculates the policy gradients with respect to the αN trajectories with the lowest returns. That is, CVaR-PG aims to follow the policy gradients corresponding to the tail distribution defined by

$$P_{\phi_0,\alpha}^{\pi_\theta}(C,\tau) = \alpha^{-1}\mathbf{1}_{R(\tau) \le q_\alpha(R|\pi_\theta)} P_{\phi_0}^{\pi_\theta}(C,\tau) \tag{4}$$

Notice that by considering only α of the trajectories, CVaR-PG essentially suffers from α^{-1}-reduction in sample efficiency in comparison to risk-neutral PG.

Proposition 1 shows that if we could sample trajectories directly from $P_{\phi_0,\alpha}^{\pi_\theta}$, we would reduce the variance of the policy gradient estimate (and thus increase the sample efficiency) back by a factor of α^{-1}. This will motivate the CEM in Section 4, which will aim to modify ϕ such that $P_\phi^{\pi_\theta} \approx P_{\phi_0,\alpha}^{\pi_\theta}$.

Proposition 1 (Variance reduction)**.** *If the quantile estimation error is negligible ($\hat{q}_\alpha = q_\alpha(R|\pi_\theta)$ in Equation (3)), then*

$$\mathrm{Var}_{\tau_i \sim P_{\phi_0,\alpha}^{\pi_\theta}}(\nabla_\theta \hat{J}_\alpha(\{\tau_i\}_{i=1}^N; \pi_\theta)) \le \alpha \cdot \mathrm{Var}_{\tau_i \sim P_{\phi_0}^{\pi_\theta}}(\nabla_\theta \hat{J}_\alpha(\{\tau_i\}_{i=1}^N; \pi_\theta)).$$

Proof sketch (see the full proof in Appendix B). Since the left term corresponds to the sample distribution $P_{\phi_0,\alpha}^{\pi_\theta}$, the corresponding IS weights are $w \equiv \alpha$ w.p. 1. When applying IS analysis to the expected value, w cancels out the distributional shift (as in Equation 5), resulting in the same expected gradient estimate. When applying the same analysis to the variance, we begin with the square weight w^2, thus a $w = \alpha$ factor still remains after the distributional shift compensation. □

The variance reduction can be connected to sample efficiency through the convergence rate as follows. According to Theorem 5.5 in Xu et al. [2020], denoting the initial parameters by θ_0, the convergence of any CVaR-PG algorithm can be written as $\mathbb{E}[\|\nabla_\theta J_\alpha(\pi_\theta)\|^2] \le \mathcal{O}(\frac{J_\alpha(\theta)-J_\alpha(\theta_0)}{M}) + \mathcal{O}(\frac{\mathrm{Var}(\nabla_\theta \hat{J}_\alpha(\{\tau_i\}_{i=1}^N;\pi_\theta))}{\alpha N})$. Clearly, variance reduction of α-factor linearly improves the second term. In particular, it cancels out the denominator's α-factor attributed to tail sub-sampling, and brings the sample efficiency back to the level of the risk-neutral PG.

4 The Cross-entropy Soft-Risk Algorithm

Algorithm 1 presents our Cross-entropy Soft-Risk algorithm (***CeSoR***), which uses a PG approach to maximize $J_\alpha(\pi_\theta)$ in (2). CeSoR adds two components on top of CVaR-PG: *soft-risk scheduling* to address the blindness to success analyzed in Section 3.1, and *CE sampling* to address the sample efficiency analyzed in Section 3.2.

Algorithm 1: CeSoR

```
1  **Input**: risk level $\alpha$; context distribution $D_\phi$; original context parameter $\phi_0$; training steps $M$; trajectories sampled per batch $N$, where $\nu$ fraction of them is from the original $D_{\phi_0}$; smoothed CE quantile $\beta$; risk-level scheduling factor $\rho$
2  **Initialize:** policy $\pi_\theta$,  $\phi \leftarrow \phi_0$,
3  $N_o \leftarrow \lfloor \nu N \rfloor$,  $N_s \leftarrow \lceil (1-\nu)N \rceil$
4  **for** $m$ *in* $1:M$ **do**
       // Sample contexts
5      Sample $\{C_{o,i}\}_{i=1}^{N_o} \sim D_{\phi_0}$,  $\{C_{\phi,i}\}_{i=1}^{N_s} \sim D_\phi$
6      $C \leftarrow (C_{o,1},\dots,C_{o,N_o},C_{\phi,1},\dots,C_{\phi,N_s})$
7      $w_{o,i} \leftarrow 1,\ \forall i \in \{1,\dots,N_o\}$
8      $w_{\phi,i} \leftarrow \frac{D_{\phi_0}(C_{\phi,i})}{D_\phi(C_{\phi,i})},\ \forall i \in \{1,\dots,N_s\}$
9      $w \leftarrow (w_{o,1},\dots,w_{o,N_o},w_{\phi,1},\dots,w_{\phi,N_s})$
       // Sample trajectories
10     $\{\tau_{C_{o,i}}\},\{\tau_{C_{\phi,i}}\} \leftarrow$ run_episodes($\pi_\theta$, $C$)
       // Update CE sampler
11     $q \leftarrow \max(\hat{q}_\alpha(\{R(\tau_{C_{o,i}})\}), \hat{q}_\beta(\{R(\tau_{C_{\cdot,i}})\}))$
12     $\phi \leftarrow \mathrm{argmax}_{\phi'} \sum_{i \le N} w_i\, \mathbf{1}_{R(\tau_{C_i}) \le q} \log D_{\phi'}(C_i)$
       // PG step (e.g., Eq. 6)
13     $\alpha' \leftarrow \max(\alpha,\, 1-(1-\alpha)\cdot m/(\rho \cdot M))$
14     $q' \leftarrow \hat{q}_{\alpha'}(\{R(\tau_{C_{o,i}})\})$
15     $\theta \leftarrow$ CVaR_PG($\pi_\theta$, $(\{\tau_{C_{o,i}}\},\{\tau_{C_{\phi,i}}\})$, $w$, $q'$)
```

Soft-risk scheduler: We set the policy optimizer (Line 15 in Algorithm 1) to use a soft risk level α' that gradually decreases from 1 to α (Line 13 and Figure 2). This is motivated by the blindness to success analyzed in Section 3.1: by modifying the risk level to $\alpha' > \alpha$, and specifically $\alpha' \approx 1$ at the beginning of training, we guarantee that there cannot be a wider tail barrier $\beta > \alpha'$. Thus, CeSoR can feed the optimizer with trajectories whose returns $q_\beta^\pi < R \le q_{\alpha'}^\pi$ are higher than any constant tail; and since the fed returns are not constant, they do not eliminate the gradient. In this sense, CeSoR looks beyond local optimization-plateaus to prevent the blindness to success.

The scheduling defined in Line 13 and Figure 2 is heuristic. As demonstrated in Section 5, once we understand the limitation of blindness to success, this simple heuristic is sufficient to bypass the blindness. An adaptive α' scheduling that maximizes blindness prevention probability would require tighter concentration inequalities [Boucheron et al., 2013], and is left for future work.

Cross Entropy Method (CEM): The CEM [de Boer et al., 2005] is a general approach to rare-event sampling and optimization, which we use to sample high-risk contexts and trajectories. First, we review the standard CEM in terms adjusted to our setting and notations (for a more general presentation, see Algorithm 2 in the appendix). Then, we discuss the limitations of the standard CEM in the RL settings, and present our dynamic, regularized version of the CEM.

Motivated by the sample efficiency analysis of Section 3.2, we wish to align the agent's experience with the α worst-case returns – by sampling contexts whose corresponding trajectory-returns are likely to be below $q_\alpha(R|\pi_\theta)$. That is, we wish to sample context-trajectory pairs from $P_{\phi_0,\alpha}^{\pi_\theta}$ of (4). To that end, the CEM searches for a value of ϕ for which $P_\phi^{\pi_\theta}$ is similar to $P_{\phi_0,\alpha}^{\pi_\theta}$. More precisely, it looks for ϕ^* that minimizes the KL-divergence (i.e., cross-entropy) between the two:

$$\begin{aligned}\phi^* &\in \mathrm{argmin}_{\phi'}\ D_{KL}\big(P_{\phi_0,\alpha}^{\pi_\theta}(C,\tau)\,||\,P_{\phi'}^{\pi_\theta}(C,\tau)\big)\\ &= \mathrm{argmax}_{\phi'}\ \mathbb{E}_{(C,\tau)\sim P_{\phi_0}^{\pi_\theta}}\big[\alpha^{-1}\mathbf{1}_{R(\tau)\le q_\alpha(R|\pi_\theta)} \log D_{\phi'}(C)\big]\\ &= \mathrm{argmax}_{\phi'}\ \mathbb{E}_{(C,\tau)\sim P_{\phi}^{\pi_\theta}}\big[\alpha^{-1}w(C,\tau)\,\mathbf{1}_{R(\tau)\le q_\alpha(R|\pi_\theta)} \log D_{\phi'}(C)\big],\end{aligned} \tag{5}$$

where $P_{\phi'}^{\pi_\theta}(C,\tau) = D_{\phi'}(C)P_C^{\pi_\theta}(\tau)$ (Section 2), and $w(C,\tau) = \frac{P_{\phi_0}^{\pi_\theta}(C,\tau)}{P_\phi^{\pi_\theta}(C,\tau)} = \frac{D_{\phi_0}(C)}{D_\phi(C)}$ is the IS weight corresponding to the sample distribution $(C,\tau) \sim P_\phi^{\pi_\theta}$. The optimization problem in Equation (5) often reduces to a simple closed-form calculation: if D_ϕ is a Gaussian, for example, ϕ^* reduces to the weighted expectation and variance of $\{C \mid R(\tau) \le q_\alpha\}_{C,\tau\sim P_\phi^{\pi_\theta}}$ with the IS weights $w(C,\tau)$.

Equation (5) may produce noisy results when estimated from data $\{(C_i,\tau_i)\}_{i=1}^N$, unless $N \gg \alpha^{-1}$, since only αN trajectory-samples satisfy $R(\tau) \le q_\alpha$ and are used in the estimation. To address this,

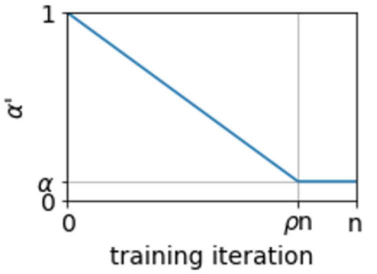


Figure 2: The soft-risk scheduling (Algorithm 1, Line 13). The linear phase $\alpha' > \alpha$ prevents the blindness to success (Section 3.1), while the CEM still preserves risk aversion. The final constant phase $\alpha' = \alpha$ provides a stationary objective and allows CeSoR to converge (Appendix C).

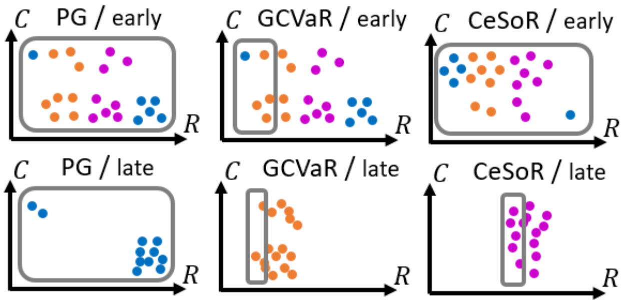


Figure 3: An illustration of training batches. Each point represents an episode with return R and context C. Points of the same color correspond to "similar" agent actions that induce similar policy gradients. Mean-PG averages over the *whole* batch and learns the blue strategy. CVaR-PG considers the *left* part (low returns) and learns the orange strategy. CeSoR over-samples the *upper* part (high-risk contexts), and only later decreases α' to explicitly focus on low returns, thus learning the purple strategy. The illustrated episodes are analogous to the strategies in Figures 1d,4.

the CEM reaches the α-tail gradually over iterations. Every iteration, it samples a batch of contexts $\{C_i\}_{i=1}^{N}$ from the current distribution D_ϕ, and then solves Equation (5) with respect to a *higher* quantile $q \geq q_\alpha$. More specifically, denote by $\hat{q}_\alpha^\phi$ the estimated α-quantile of $\{R(\tau)\}_{C,\tau\sim P_\phi^{\pi_\theta}}$; then, we set $q = \max(\hat{q}_\alpha^{\phi_0},\, \hat{q}_\beta^{\phi})$ with a hyperparameter $\beta > \alpha$ (often $\beta = 0.2$). Since the data is drawn from $P_\phi^{\pi_\theta}$, this guarantees at least βN samples per update step. The quantile $\hat{q}_\alpha^{\phi_0}$ corresponds to the α-tail of the original context-distribution, and can be viewed as a stopping condition: once $\hat{q}_\alpha^{\phi_0} > \hat{q}_\beta^{\phi}$, many of our samples are already in the tail, and β is no longer needed to smooth the update of ϕ.

Dynamic-target CEM: The standard CEM assumes to search for the tail of a *constant* distribution. In our setting, however, we look for the tail of the distribution of the returns $R(\tau)$, where $C, \tau \sim P_{\phi_0}^{\pi_\theta}$ depend on π_θ and thus are non-stationary throughout the training. The non-stationarity poses several challenges for the CEM. First, the stopping condition $\hat{q}_\alpha^{\phi_0}$ varies with π_θ and has to be re-estimated every iteration[1]. Second, the high-risk contexts C (which correspond to the lowest returns) may vary as the agent evolves; and if the CEM learns to only sample a strict subset of the context space, then it may miss such changes in the high-risk contexts.

We address both issues using reference samples: every iteration, we sample *two* batches of contexts – $\{C_{\phi,i}\}_{i=1}^{N_s}$ from the current context distribution D_ϕ and $\{C_{o,i}\}_{i=1}^{N_o}$ from the original distribution D_{ϕ_0}. The reference contexts provide an important regularization: they guarantee continual exposure to the whole context space, in case that the high-risk contexts vary. In addition, the reference samples were empirically found to stabilize the estimation of $\hat{q}_\alpha^{\phi_0}$ (Line 11 in Algorithm 1).

Consider the two batches of context-trajectory pairs, and denote the estimated return quantile $\hat{q}_\alpha = \hat{q}_\alpha(\{R(\tau_i)\}_{i=1}^{N_o})$. We can estimate the CVaR policy gradient, using the notation $\forall 1 \leq i \leq N_o + N_s$: $C_i = \begin{cases} C_{o,i} & \text{if } 1 \leq i \leq N_o \\ C_{\phi,i-N_o} & \text{if } N_o + 1 \leq i \leq N_o + N_s \end{cases}$, by

$$\nabla_\theta \hat{J}_\alpha(\pi_\theta) = \frac{1}{\alpha(N_o + N_s)} \sum_{i=1}^{N_o+N_s} w_i \cdot \mathbf{1}_{R(\tau_i)\leq \hat{q}_\alpha} \left(R(\tau_i) - \hat{q}_\alpha\right) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}; s_{i,t}), \tag{6}$$

where $w_i = 1$ for $1 \leq i \leq N_o$ and $w_i = D_{\phi_0}(C_i)/D_{\phi^*}(C_i)$ for $N_o + 1 \leq i \leq N_o + N_s$.

Note that if the policy learning scale is slower than that of ϕ, the target context distribution $P_{\phi_0,\alpha}^{\pi_\theta}$ is effectively stationary in the ϕ-optimization problem. In that case, according to de Mello and Rubinstein [2003], the CEM will converge to the KL-divergence minimizer ϕ^* of (5).

[1] For the sake of coherent notations, we presented the CEM with the quantile objective $q_\alpha(R|\pi_\theta)$. In fact, the standard CEM is usually defined with a constant numeric objective $q_0 \in \mathbb{R}$ rather than a quantile; hence, as shown in Algorithm 2 in the appendix, the standard CEM does not require any quantile estimation at all.

Sample efficiency in practice: Proposition 1 guarantees an α^{-1}-increase in sample efficiency when using an accurate quantile estimate $\hat{q}_\alpha = q_\alpha(R|\pi_\theta)$ and sampling exactly from $P^{\pi_\theta}_{\phi_0,\alpha}$. The latter condition is equivalent to the CE-sampler reaching its objective $D_{KL}(P^{\pi_\theta}_{\phi_0,\alpha} \,||\, P^{\pi_\theta}_{\phi}) = 0$. In practice, $P^{\pi_\theta}_{\phi_0,\alpha}$ can only be approximated, and the sample efficiency is increased – but by a smaller factor than α^{-1}. Appendix D.3 demonstrates the increased sample size exploited by CeSoR in our experiments.

If $\hat{q}_\alpha \neq q_\alpha(R|\pi_\theta)$, the quantile estimation error may theoretically lead to unbounded IS weights (see Appendix B). Practically, we address this by clipping the weights (as mentioned in Section 5), and by constraining the family of permitted distributions $\{D_\phi\}_\phi$ to have a constant support independently of ϕ. A side-effect is a function approximation error of the family $\{D_\phi\}_\phi$, as $D_{\phi^*}(C)P^{\pi_\theta}_C(\tau)$ cannot replicate the tail distribution $P^{\pi_\theta}_{\phi_0,\alpha}(C,\tau)$ to achieve the full α^{-1}-increase in sample efficiency.

Another limitation in the expressiveness of $P^{\pi_\theta}_{\phi}(C,\tau) = D_\phi(C)P^{\pi_\theta}_C(\tau)$ occurs when the context C only controls part of the environment randomness in $P^{\pi_\theta}_C(\tau)$. As an extreme example in the Guarded Maze, after π_θ already learns to avoid the short path, the context (guard cost) does not affect the outcome at all anymore. Indeed, Figure 10a in the appendix shows that high guard costs are sampled in the beginning; then, once the short path is avoided, the sampler gradually falls back to the original context distribution. Note that in this example, the invariance to C began after the learning was essentially done, hence the CEM did play its part effectively.

Finally, note that the soft risk creates an intentional bias in the gradient estimate (to overcome the blindness to success). As a result, in the first phase of training ($\alpha' \gg \alpha$), only a few trajectories are overlooked every iteration. As α' approaches α, the number of overlooked trajectories increases, and so is the importance of over-sampling the tail. In the final steady-state phase ($\alpha' = \alpha$), the sample inefficiency is most severe, the soft risk produces no further biases, and the CEM helps CeSoR to reduce the high variance in the policy gradient estimation.

The harmony between the soft risk and the CEM: Soft risk has the inherent side effect of reducing the risk aversion. In the Guarded Maze, for example, as demonstrated in Section 5.1, soft risk alone leads to learning the short path (instead of the risk-averse long path). Fortunately, the CEM reduces this side effect. In that sense, the two mechanisms complement each other: $\alpha' > \alpha$ allows the *optimizer* to learn policies with high returns, while the CE *sampler* still preserves the risk aversion – as illustrated in Figure 3. This connection stands in addition to the independent motivations of the two mechanisms, as discussed above.

Baseline optimizer: CeSoR can be implemented on top of any CVaR-PG method as a baseline (Line 15). We use the standard GCVaR [Tamar et al., 2015b], which guarantees asymptotic convergence under certain regularity conditions. Appendix C shows that these guarantees hold for CeSoR as well, when implemented on top of GCVaR. Other CVaR-PG baselines can also be used, such as the TRPO-based algorithm of Rajeswaran et al. [2017]. However, such methods often include heuristics that introduce additional gradient estimation bias (to reduce variance), and thus do not necessarily guarantee the same theoretical convergence.

5 Experiments

We conduct experiments in 3 different domains. We implement **CeSoR** on top of a standard CVaR-PG method, which is also used as a risk-averse baseline for comparison. Specifically, we use the standard **GCVaR** baseline [Tamar et al., 2015b], which guarantees convenient convergence properties (see Appendix C) and is simple to implement and analyze. We also use the standard policy gradient (**PG**) as a risk-neutral baseline. We stress that the comparison to PG is only intended to present the mean-CVaR tradeoff, while each method legitimately optimizes its own objective. Appendix H also compares CeSoR to risk-neutral and risk-averse Distributional RL algorithms.

In all the experiments, all agents are trained using Adam [Diederik P. Kingma, 2014], with a learning rate selected manually per benchmark and $N = 400$ episodes per training step. Every 10 steps we run validation episodes, and we choose the final policy according to the best validation score (best mean for PG, best CVaR for GCVaR and CeSoR). For CeSoR, unless specified otherwise, $\nu = 20\%$ of the trajectories per batch are drawn from the original distribution D_{ϕ_0}; $\beta = 20\%$ are used for the CE update; and the soft risk level reaches α after $\rho = 80\%$ of the training. As mentioned in Section 4, for numerical stability, we also clip the IS weights (Algorithm 1, Line 9) to the range $[1/5, 5]$.

Every policy is modeled as a neural network with $tanh$ activation on its middle layers and $softmax$ operator on its output, with temperature 1 in training (i.e., network outputs are actions probabilities), and 0 in validation and test (i.e., the max output is the selected action). We use a middle layer with 32 neurons in Section 5.2, 16 neurons in Section 5.3, and no middle layer (linear model) in Section 5.1.

In each of the 3 domains, the experiments required a running time of a few hours on an Ubuntu machine with eight i9-10900X CPU cores. In addition to these RL-related experiments, Appendix D presents dedicated experiments for the independent CE module.

5.1 The Guarded Maze

Benchmark: The Guarded Maze benchmark is defined in Section 1. For the experiments, we set a target risk level of $\alpha = 0.05$, and train each agent for $n = 250$ steps with the parameters described above. The CEM controls C through $\phi = (\phi_1, \phi_2)$, where $\phi_0 = (0.2, 32)$ as mentioned above, and updates ϕ_1, ϕ_2 using the weighted means of C_1 and C_2, respectively. As an ablation test, we add two partial variants of our CeSoR: **CeR** (with CE, without α-scheduling) and **SoR** (with scheduling, without CE). See more details in Appendix E.1.

Results: Figure 1a summarizes the test scores, and Figure 1d illustrates a sample episode. PG learned the short path, maximizing the average but at the cost of poor returns whenever charged by the guard. CeSoR, on the other hand, successfully learned to follow the CVaR-optimal long path. GCVaR, which also aimed to maximize the CVaR, failed to do so. As analyzed in Figure 4, throughout GCVaR training, the agent takes the long path in up to 50% of the episodes per batch, but none of these episodes is ever included in the bottom $\alpha = 5\%$ that are fed to the optimizer. Thus, GCVaR is entirely *blind* to the successful episodes and fails to learn the corresponding strategy. In fact, in most training steps, *all* the worst episodes of GCVaR reach neither the guard nor the target, leading to a constant return of -32, a tail barrier, and a zero loss-gradient.

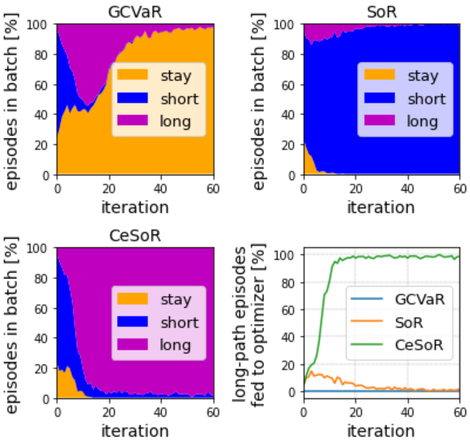


Figure 4: GCVaR, SoR, CeSoR: %-episodes that did not reach the target ("stay"), or reached it through the short or the long path in the Guarded Maze. Bottom Right: %-long-paths among the trajectories fed to the optimizer. See more details in Figure 13.

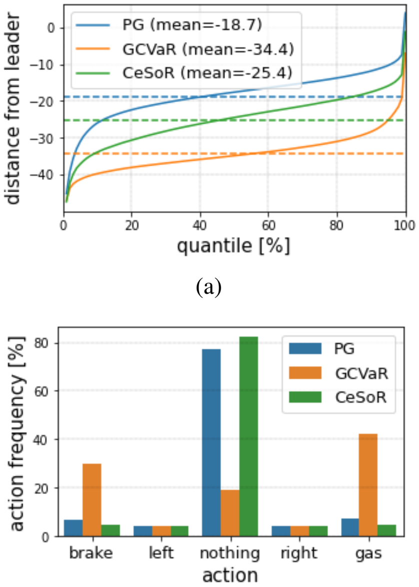
CeR suffers from blindness to success just as GCVaR. SoR is exposed to the successful long-path episodes thanks to soft risk scheduling; however, due to the reduced risk-aversion, it fails to prefer the long path over the short one. Only CeSoR both observes the *"good" strategy* (thanks to soft risk scheduling) and judges it under *"bad" environment variations* (thanks to the CEM). Appendix E.2 presents a detailed analysis of the learning dynamics, the blindness to success and the learned policies. It is important to notice that standard optimization tweaks cannot bring GCVaR to learn the long path: a "warm-start" from a standard PG only encourages the short-path policy (as in SoR); and increased batch size N does not expose the optimizer to the long path (see Theorem 1).

5.2 The Driving Game

Benchmark: The Driving Game is based on an inverse-RL benchmark used by Majumdar et al. [2017] and Singh et al. [2018]. The agent's car has to follow the leader (an "erratic driver") for 30 seconds as closely as possible without colliding. Every 1.5 seconds (i.e., 20 times per episode), the leader chooses a random action (independently of the agent): drive straight, accelerate, decelerate, change lane, or brake hard ("emergency brake"), with respective probabilities $\phi_0 = (0.35, 0.3, 0.248, 0.1, 0.002)$. We denote the sequence of leader actions by $C \in \{1, ..., 5\}^{20}$.

Every 0.5 seconds (60 times per episode), the agent observes its relative position and velocity to the leader, with a delay of 0.7 seconds (representing reaction time), as well as its own acceleration and steering direction. The agent chooses one of the five actions: drive in the same steering direction, accelerate, decelerate, turn left, or turn right. Changing lane is not an atomic action and has to be learned using turns. The rewards express the requirements to stay behind the leader, on the road, on the same lane, not too far behind and without colliding. See the complete details in Appendix F.1.

We set $\alpha = 0.01$, and train each agent for $n = 500$ steps. To initiate learning, for each agent we begin with shorter training episodes of 6 seconds and gradually increase their length. The CEM controls the leader's behavior through the probabilities $\phi = \{\phi_i\}_{i=1}^{5}$ described above.



(a)

(b)

Figure 5: Over all the time-steps in all the test episodes in the Driving Game, the distribution of (a) the distance between the agent and the leader, (b) the agent actions. Evidently, CeSoR learns to keep more distance than the risk-neutral PG, and has a slightly less frequent use of the gas and the brake.

Results: Figure 1b summarizes the test scores of the agents, where CeSoR presents a reduction of 28% in the CVaR cost in comparison to the baselines. GCVaR completely fails to learn a reasonable policy – losing in terms of CVaR even to the risk-neutral PG. Figure 5 shows that CeSoR learned an arguably-intuitive policy for risk averse driving: it keeps a safer distance from the leader, and uses the gas and the brake less frequently. This results in complete avoidance of the rare accidents occurring to PG, as demonstrated in Figure 1e. In Appendix D, we also see that by over-sampling turns and emergency brakes of the leader, the CEM manages to align the mean return of the training samples with the 1%-CVaR of the environment, and significantly increases the data efficiency.

5.3 The Computational Resource Allocation Problem

Benchmark: Computational resource allocation in serving systems, and in particular the tradeoff between resource cost and serving latency, is an important challenge to both academia [Jiang et al., 2013, Tessler et al., 2022] and industry [Barr, 2018, Lunden, 2022]. In popular applications such as E-commerce and news, latency is most critical at times of peak loads [Garces, 2019], making CVaR a natural metric for risk-averse optimization. In our benchmark, the agent allocates servers to handle user requests, managing the tradeoff between servers cost and time-to-service (TTS). Requests arrive randomly with a constant rate, up to rare events that cause sudden peak loads, whose frequency is controlled by the CE sampler. See Appendix G for more details.

Results: As shown in Figure 1c, CeSoR significantly improves the CVaR return, and does not compromise the mean as much as GCVaR. As demonstrated in Figure 1f, CeSoR learned to allocate a default of 5 servers and react to peak loads as needed, whereas GCVaR simply allocates 8 servers at all times. PG only allocates 4 servers by default, and thus its TTS is more sensitive to peak loads. Appendix G describes the complete implementation and detailed results, discusses the poor parameterization of D_ϕ in this problem and shows the robustness of CeSoR to that parameterization.

6 Summary and Future Work

We introduced CeSoR, a novel method for risk-averse RL, focused on efficient sampling and soft risk. In a variety of experimental domains, in comparison to a risk-averse baseline, CeSoR demonstrated higher CVaR metric, better sample-efficiency, and elimination of blindness to success – where the latter two were also analyzed theoretically.

There are certain limitations to CeSoR. First, we assume to have at least partial control over the training conditions, through a parametric family of distributions that needs to be selected. Second, CeSoR can be applied robustly on top of any CVaR-PG method, but is currently not applicable to non-PG methods. Since the limitations of CVaR-PG apply in other risk-averse methods as well (as we demonstrated for Distributional RL), future work may adjust CeSoR to such methods, as well as to other risk measures. Third, in terms of blindness to success and estimation variance, CeSoR shows both theoretical and empirical improvement – but is not proven optimal. Future work may look for optimal design of CEM or risk scheduling. Considering the current results and the potential extensions, we believe CeSoR may open the door for more practical applications of risk-averse RL.

Acknowledgements: This research was supported by the Israel Science Foundation (grant 2199/20).

References

Jeff Barr. Predictive scaling for EC2, powered by machine learning, 2018. URL https://aws.amazon.com/blogs/aws/new-predictive-scaling-for-ec2-powered-by-machine-learning/.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

Cristian Bodnar, Adrian Li, Karol Hausman, Peter Pastor, and Mrinal Kalakrishnan. Quantile QT-Opt for risk-aware vision-based robotic grasping. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.075.

V. S. Borkar and S. P. Meyn. Risk-sensitive optimal control for Markov decision processes with monotone cost. *Mathematics of Operations Research*, 27(1):192–209, 2002.

Vivek Borkar and Rahul Jain. Risk-constrained Markov decision processes. *IEEE Transactions on Automatic Control*, 2014.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

Y. Chow and M. Ghavamzadeh. Algorithms for CVaR optimization in MDPs. In *Proceedings of Advances in Neural Information Processing Systems 27*, pages 3509–3517, 2014.

Y. Chow, A. Tamar, S. Mannor, and M. Pavone. Risk-sensitive and robust decision-making: a CVaR optimization approach. In *Advances in Neural Information Processing Systems*, 2015.

Sebastian Curi, Kfir Y. Levy, Stefanie Jegelka, and Andreas Krause. Adaptive sampling for stochastic risk-averse learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1036–1047. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0b6ace9e8971cf36f1782aa982a708db-Paper.pdf.

Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1096–1105, 2018a.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018b. doi: 10.1609/aaai.v32i1.11791. URL https://ojs.aaai.org/index.php/AAAI/article/view/11791.

Frederic Dambreville. Cross-entropy method: convergence issues for extended implementation, 2006.

P. T. de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134:19–67, 2005.

Tito Homem de Mello and Reuven Y. Rubinstein. Rare event estimation for static models via cross-entropy and importance sampling, 2003.

Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Yingjie Fei, Zhuoran Yang, Yudong Chen, and Zhaoran Wang. Exponential bellman equation and improved regret bounds for risk-sensitive reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Carla Prat Garces. The problem of peak loads in web applications and its solutions, 2019.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

Mathieu Godbout, Maxime Heuillet, Sharath Chandra, Rupali Bhati, and Audrey Durand. CARL: Conditional-value-at-risk adversarial reinforcement learning. *arXiv preprint arXiv:2109.09470*, 2021.

Ido Greenberg. Cross entropy method with non-stationary score function. `https://pypi.org/project/cross-entropy-method/`, 2022.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Takuya Hiraoka, Takahisa Imagawa, Tatsuya Mori, Takashi Onishi, and Yoshimasa Tsuruoka. Learning robust options by conditional value at risk optimization. *NeurIPS*, 05 2019.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

L. Jeff Hong and Guangwu Liu. Simulating sensitivities of conditional value at risk. *Management Science*, 55(2):281–293, 2009. ISSN 00251909, 15265501. URL `http://www.jstor.org/stable/40539145`.

Audrey Huang, Liu Leqi, Zachary C. Lipton, and Kamyar Azizzadenesheli. On the convergence and optimality of policy gradient for markov coherent risk, 2021a. URL `https://arxiv.org/abs/2103.02827`.

Audrey Huang, Liu Leqi, Zachary C. Lipton, and Kamyar Azizzadenesheli. On the convergence and optimality of policy gradient for markov coherent risk, 2021b. URL `https://arxiv.org/abs/2103.02827`.

Kevin Huang, Sahin Lale, Ugo Rosolia, Yuanyuan Shi, and Anima Anandkumar. CEM-GD: Cross-entropy method with gradient descent planner for model-based reinforcement learning, 2021c.

Jing Jiang, Jie Lu, Guangquan Zhang, and Guodong Long. Optimal cloud resource auto-scaling for web applications. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 58–65. IEEE, 2013.

Ramtin Keramati, Christoph Dann, Alex Tamkin, and Emma Brunskill. Being optimistic to be conservative: Quickly learning a CVaR policy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:4436–4443, 04 2020.

Leslie Kish. *Survey Sampling*. New York: John Wiley and Sons, Inc., 1965.

Tom Leinster. Effective sample size, 2014.

Ingrid Lunden. Intel confirms acquisition of AI-based workload optimization startup granulate, reportedly for up to $650M, 2022.

Anirudha Majumdar, Sumeet Singh, Ajay Mandlekar, and Marco Pavone. Risk-sensitive inverse reinforcement learning via coherent risk models. *Robotics: Science and Systems*, 07 2017. doi: 10.15607/RSS.2017.XIII.069.

Shie Mannor, Reuven Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. *Proceedings, Twentieth International Conference on Machine Learning*, 2, 07 2003.

Cosmin Paduraru, Daniel J. Mankowitz, Gabriel Dulac-Arnold, Jerry Li, Nir Levine, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning Journal*, 2021.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2817–2826. JMLR.org, 2017.

L.A. Prashanth and M. Ghavamzadeh. Actor-critic algorithms for risk-sensitive MDPs. In *Proceedings of Advances in Neural Information Processing Systems 26*, pages 252–260, 2013.

L.A. Prashanth and M. Ghavamzadeh. Variance-constrained actor-critic algorithms for discounted and average reward MDPs. *Machine Learning Journal*, 105(3):367–417, 2016.

Wei Qiu, Xinrun Wang, Runsheng Yu, Xu He, R. Wang, Bo An, Svetlana Obraztsova, and Zinovi Rabinovich. RMIX: Learning risk-sensitive policies for cooperative reinforcement learning agents. *ArXiv*, abs/2102.08159, 2021.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *ICLR*, 2017.

Makoto Sato, Hajime Kimura, and Syumpei Kobayashi. TD algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of The Japanese Society for Artificial Intelligence*, 16:353–362, 2001.

Sumeet Singh, Jonathan Lacotte, Anirudha Majumdar, and Marco Pavone. Risk-sensitive inverse reinforcement learning via semi- and non-parametric methods. *The International Journal of Robotics Research*, 37, 04 2018. doi: 10.1177/0278364918772017.

Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Policy gradient for coherent risk measures. *NIPS*, 2015a.

Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. *AAAI'15*, page 2993–2999, 2015b.

Yichuan Tang, Jian Zhang, and Ruslan Salakhutdinov. Worst cases policy gradients. In *CoRL*, 2019.

Chen Tessler, Yuval Shpigelman, Gal Dalal, Amit Mandelbaum, Doron Haritan Kazakov, Benjamin Fuhrer, Gal Chechik, and Shie Mannor. Reinforcement learning for datacenter congestion control. *SIGMETRICS Perform. Eval. Rev.*, 49(2):43–46, jan 2022. ISSN 0163-5999. doi: 10.1145/3512798.3512815. URL `https://doi.org/10.1145/3512798.3512815`.

J. R. R. Tolkien. *The Lord of the Rings: The Fellowship of the Ring*. George Allen and Unwin, 1954.

Nithia Vijayan and L. A. Prashanth. Likelihood ratio-based policy gradient methods for distorted risk measures: A non-asymptotic analysis. *ArXiv*, abs/2107.04422, 2021.

Edoardo Vittori, Michele Trapletti, and Marcello Restelli. Option hedging with risk averse reinforcement learning. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375849. doi: 10.1145/3383455.3422532. URL `https://doi.org/10.1145/3383455.3422532`.

T. Xie, B. Liu, Y. Xu, M. Ghavamzadeh, Y. Chow, D. Lyu, and D. Yoon. A block coordinate ascent algorithm for mean-variance optimization. In *Proceedings of Advances in Neural Information Processing Systems 232*, pages 1073–1083, 2018.

Pan Xu, Felicia Gao, and Quanquan Gu. An improved convergence analysis of stochastic variance-reduced policy gradient. In *Uncertainty in Artificial Intelligence*, pages 541–551. PMLR, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the summary (Section 6).
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Risk sensitive RL is an abstract task and is not specifically associated with any negative-impact applications.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] For each result, Section 3 either states the assumptions directly or refers to Appendices A-C.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3 with references to Appendices A-C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] A repository of our code presents Jupyter notebooks that reproduce the experimental results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] General details in the beginning of Section 5, and specific details per experiment in Sections 5.1-5.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In Figures 1,11,16,20a, we provide not only the average return but the full distribution, which implicitly includes the information of error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the beginning of Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] An anonymized repository of our code is linked from the abstract.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Contents

A Blindness to Success: Proof of Theorem 1

Theorem 1 considers the probabilistic event of a global blindness to success over n consecutive training steps. We begin with a local blindness in a single training step.

Lemma 1 (Local blindness to success)**.** Let a risk level $\alpha \in (0,1)$ and a CVaR-PG training step $m \geq 1$, and let $\beta \in (\alpha, 1)$. Denote $A = \left\{ \{\tau_{m,i}\}_{i=1}^N \in \mathcal{T}^N \;\middle|\; q_\beta^{\pi_m} < \hat{q}_\alpha(\{R(\tau_{m,i})\}_{i=1}^N) \right\}$. Then,

$$\mathbb{P}\left(A\right) \leq e^{-\frac{N(\beta-\alpha)^2}{2\beta(1-\beta)}} \leq e^{-2N(\beta-\alpha)^2}$$

Proof. Denote $R_i = R(\tau_{m,i})$, $\chi_i^q = \mathbf{1}_{R_i > q}$, and $\chi_i = \chi_i^{q_\beta^{\pi_m}}$. Note that $\chi_i \sim \mathit{Bernoulli}(1 - \beta)$. Also denote the percent of high-return trajectories by $n_q = \sum_{i=1}^N \chi_i^q / N$ and $n^* = n_{q_\beta^\pi}$. Since $\hat{q}_\alpha(\{R_i\}_{i=1}^N) = \min\left\{q \,\middle|\, \frac{|\{i \,|\, R_i \leq q\}|}{N} \geq \alpha\right\} = \min\left\{q \,\middle|\, \frac{|\{i \,|\, R_i > q\}|}{N} < 1-\alpha\right\} = \min\left\{q \,\middle|\, \frac{1}{N}\sum_{i=1}^N \chi_i^q < 1-\alpha\right\}$, we have $q_\beta^\pi < \hat{q}_\alpha(\{R_i\}) \Leftrightarrow n^* \geq 1-\alpha$, i.e., $A = (n^* \geq 1-\alpha)$.

Since $\mathbb{P}(0 \leq \chi_i \leq 1) = 1$, $E\left[\chi_i\right] = 1-\beta$ and the Bernoulli χ_i are sub-Gaussian with variance factor $\sigma^2 = 1/4$, by Hoeffding inequality [Hoeffding, 1994] we obtain

$$\mathbb{P}(A) = \mathbb{P}(n^* \geq 1-\alpha) = \mathbb{P}(n^* - E\left[n^*\right] \geq \beta - \alpha) \leq e^{-\frac{N^2(\beta-\alpha)^2}{2\sum_i 1/4}} = e^{-2N(\beta-\alpha)^2}.$$

□

Note that Lemma 1 does not depend on a tail-barrier: it simply implies that since a CVaR-PG algorithm focuses on the worst α trajectories in every batch, we do not expect trajectories with high returns $R(\tau_{m,i}) > q_\beta^{\pi_m}$ to be fed to the optimizer. Still, in general, even if high-return trajectories are ignored, the CVaR-PG can learn to avoid low-return trajectories with $R(\tau_{m,i}) < q_\beta^{\pi_m}$. The tail barrier prevents this learning, since there are no returns strictly lower than $q_\beta^{\pi_m}$ – all the tail identically equals $q_\beta^{\pi_m}$. Since there are no worse trajectories to learn from, and better trajectories are ignored, this brings the training to a deadlock, as stated by Theorem 1.

Proof of Theorem 1 (stated in Section 3.1). All probabilities below are conditioned on the event of π_{m_0} having a β-tail barrier. Thus, we simplify the notation to $\mathbb{P}(\cdot) = \mathbb{P}\left(\cdot \,\middle|\, \pi_{m_0} \text{ has } \beta\text{-tail barrier}\right)$.

Denote by $\mathcal{S} = \left\{\left(\{\tau_{m,i}\}_{i=1}^N, \pi_m\right)\right\}_{m=m_0}^{m_0+n-1} \in (\mathcal{T}^N \times \Pi)^n$ the sequence of trajectory batches and policies, and by $R_m = \{R(\tau_{m,i})\}_{i=1}^N$ the returns on step m. Also denote for simplicity $\mathcal{B} = \mathcal{B}_{\alpha,\beta}^{m_0,n}$. We are interested in the probability of the event that there is no global blindness (Definition 2):

$$\neg\mathcal{B} = \neg\mathcal{B}_{\alpha,\beta}^{m_0,n} = \left\{\mathcal{S} \,\middle|\, \exists m_0 \leq m < m_0+n : \, q_\beta^{\pi_{m_0}} < \hat{q}_\alpha(R_m)\right\}.$$

Define the event of blindness at step m, along with an unchanged policy: $A_m = \left\{\mathcal{S} \,\middle|\, \pi_m = \pi_{m_0} \wedge \hat{q}_\alpha(R_m) \leq q_\beta^{\pi_{m_0}}\right\}$. Note that $\bigcap_{m=m_0}^{m_0+n-1} A_m \subseteq \mathcal{B}$, hence

$$\mathbb{P}(\neg\mathcal{B}) \leq 1 - \mathbb{P}\left(\bigcap_{m=m_0}^{m_0+n-1} A_m\right) = 1 - \prod_{m=m_0}^{m_0+n-1} \mathbb{P}\left(A_m | A_{m_0}, ..., A_{m-1}\right).$$

Thus, to complete the proof, we show below that $\mathbb{P}\left(A_m | A_{m_0}, ..., A_{m-1}\right) \geq 1-\delta$, where $\delta = e^{-2N(\beta-\alpha)^2}$, hence $\mathbb{P}(\neg\mathcal{B}) \leq 1-(1-\delta)^n \leq 1-(1-n\delta) = n\delta$.

For $m = m_0$, we have immediately $\pi_m = \pi_{m_0}$, and from Lemma 1 $\mathbb{P}(q_\beta^{\pi_{m_0}} < \hat{q}_\alpha(R_{m_0})) \leq \delta$. For $m_0+1 \leq m \leq m_0+n-1$, assume that $A_{m_0}, ..., A_{m-1}$ hold. In particular, $\hat{q}_\alpha(R_{m-1}) \leq q_\beta^{\pi_{m_0}}$, $\pi_{m-1} = \pi_{m_0}$ and π_{m-1} has a β-tail barrier. Now consider the $m-1$ training batch: for every trajectory $1 \leq i \leq N$, if $R_{m-1,i} > \hat{q}_\alpha(R_{m-1})$, then $\mathbf{1}_{R_{m-1,i} \leq \hat{q}_\alpha(R_{m-1})} = 0$; otherwise, $R_{m-1,i} \leq \hat{q}_\alpha(R_{m-1}) \leq q_\beta^{\pi_{m_0}}$, that is, $R_{m-1,i} = q_{\beta'}^{\pi_{m_0}}$ for some $\beta' \leq \beta$, and by the barrier property $R_{m-1,i} = q_\beta^{\pi_{m_0}}$ and thus $R_{m-1,i} - \hat{q}_\alpha(R_{m-1}) = 0$. Hence, the gradient in Equation (3) is 0, the policy update vanishes, and we obtain $\pi_m = \pi_{m-1} = \pi_{m_0}$. Then again, according to Lemma 1 (and since R_m and R_{m_0} are drawn from the same distribution corresponding to $\pi_m = \pi_{m_0}$), we have $\mathbb{P}(q_\beta^{\pi_{m_0}} < \hat{q}_\alpha(R_m)) = \mathbb{P}(q_\beta^{\pi_{m_0}} < \hat{q}_\alpha(R_{m_0})) \leq \delta$, as required. □

Note that the factor n may become quite negligible when the barrier is wider than α: if $n = 10^6, \alpha = 0.05, \beta = 0.25, N = 400$, for example, we still have $\mathbb{P}(\neg\mathcal{B}_{\alpha,\beta}^{m_0,n}) < 10^{-7}$. Indeed, the blindness occurs with significantly smaller barriers than the $\beta = 0.9$ demonstrated in the Guarded Maze in Appendix E.2. Note that the momentum term of the Adam algorithm [Diederik P. Kingma, 2014], while preventing the policy update from completely vanishing, was empirically insufficient to overcome the barrier in the Guarded Maze. This should not come as a surprise, since the momentum comes from previous gradients that *encouraged* the strategies of the barrier and brought them into the tail in the first place.

B Variance Reduction: Proof of Proposition 1

Proof. Define $H(C,\tau) = \alpha^{-1}\mathbf{1}_{R(\tau)\leq\hat{q}_\alpha}\left(R(\tau) - \hat{q}_\alpha\right)\nabla_\theta \sum_t \log\pi_\theta(a_t; s_t)$, such that the CVaR PG can be written as $\nabla_\theta \hat{J}_\alpha\left(\{C_i,\tau_i\}_{i=1}^N; \pi_\theta\right) = \frac{1}{N}\sum_{i=1}^N w(C_i,\tau_i)H(C_i,\tau_i)$, where $w(C,\tau) = \frac{P_{\phi_0}^{\pi_\theta}(C,\tau)}{P_{\phi_0,\alpha}^{\pi_\theta}(C,\tau)}$ is the IS weighting that accounts for the modified sample distribution. Since $C,\tau \sim P_{\phi_0,\alpha}^{\pi_\theta}$, we have $R(\tau) \leq q_\alpha(R|\pi_\theta)$ almost surely; and along with the assumption $\hat{q}_\alpha = q_\alpha(R|\pi_\theta)$, we obtain

$$w(C,\tau) = \frac{P_{\phi_0}^{\pi_\theta}(C,\tau)}{P_{\phi_0,\alpha}^{\pi_\theta}(C,\tau)} = \frac{P_{\phi_0}^{\pi_\theta}(C,\tau)}{\alpha^{-1}\mathbf{1}_{R(\tau)\leq q_\alpha(R|\pi_\theta)}P_{\phi_0}^{\pi_\theta}(C,\tau)} = \alpha.$$

The assumption $\hat{q}_\alpha = q_\alpha(R|\pi_\theta)$, when applied to Equation (3), also guarantees that $\nabla_\theta\hat{J}_\alpha$ is an unbiased gradient estimator for both sample distributions $P = P_{\phi_0}^{\pi_\theta}$ and $P = P_{\phi_0,\alpha}^{\pi_\theta}$: $\mathbb{E}_{C_i,\tau_i\sim P}[\nabla_\theta\hat{J}_\alpha\left(\{C_i,\tau_i\}_{i=1}^N;\, \pi_\theta\right)] = \nabla_\theta J_\alpha\left(\pi_\theta\right)$. Its variance over N i.i.d samples is $\text{Var}_{C_i,\tau_i\sim P}[\nabla_\theta\hat{J}_\alpha\left(\{C_i,\tau_i\}_{i=1}^N;\, \pi_\theta\right)] = \frac{1}{N}\text{Var}_{C,\tau\sim P}[\nabla_\theta\hat{J}_\alpha\left(C,\tau;\, \pi_\theta\right)]$. Denoting $g \coloneqq \nabla_\theta J_\alpha\left(\pi_\theta\right)$, we obtain:

$$\begin{aligned}
&\text{Var}_{C,\tau\sim P_{\phi_0,\alpha}^{\pi_\theta}}[\nabla_\theta\hat{J}_\alpha\left(C,\tau;\, \pi_\theta\right)] \\
=&\mathbb{E}_{C,\tau\sim P_{\phi_0,\alpha}^{\pi_\theta}}[w(C,\tau)^2 H(C,\tau)^2] - g^2 \\
=&\mathbb{E}_{C,\tau\sim P_{\phi_0}^{\pi_\theta}}[w(C,\tau) H(C,\tau)^2] - g^2 \\
=&\alpha\cdot\mathbb{E}_{C,\tau\sim P_{\phi_0}^{\pi_\theta}}[H(C,\tau)^2] - g^2 \\
\leq&\alpha\cdot(\mathbb{E}_{C,\tau\sim P_{\phi_0}^{\pi_\theta}}[H(C,\tau)^2] - g^2) \\
=&\alpha\cdot\text{Var}_{C,\tau\sim P_{\phi_0}^{\pi_\theta}}[\nabla_\theta\hat{J}_\alpha\left(C,\tau;\, \pi_\theta\right)],
\end{aligned}$$

which completes the proof. □

Note that if $\hat{q}_\alpha \neq q_\alpha(R|\pi_\theta)$, the term $\mathbf{1}_{R(\tau)\leq\hat{q}_\alpha}$ in the denominator may vanish and the IS weight $w(\tau,C)$ may become unbounded. To overcome this issue when using our CE-sampler (described in Section 4), we constrain the family of distributions $\{P_\phi^{\pi_\theta}\}_\phi$ such that the sample distribution $P_\phi^{\pi_\theta}$ always has the same support as the original distribution $P_{\phi_0}^{\pi_\theta}$ (even though this eliminates the possibility of an exact tail sampling $P_\phi^{\pi_\theta} = P_{\phi_0,\alpha}^{\pi_\theta}$). In addition, in the experiments of Section 5 we clip the IS weights directly.

C Gradient Estimation Bias and CeSoR Convergence

The gradient estimator of Equation (3) is biased due to the biasedness of the empirical quantile. However, Tamar et al. [2015b] show that the gradient estimator is still consistent, and bound its bias by $\mathcal{O}(N^{-1/2})$. Lemma 2 below proves that a similar result holds for CeSoR – despite the CEM and the risk scheduling. Given Lemma 2, CeSoR's convergence is a direct application of Theorem 5 in Tamar et al. [2015b], as stated below. The soft-risk scheduling α' introduces additional transient bias to the CVaR gradient estimate when $\alpha' > \alpha$, but this bias vanishes in the last steady-state $1-\rho$ steps when $\alpha' = \alpha$; hence, we can safely assume consistency of CeSoR's gradient estimate, and focus our asymptotic convergence analysis on the steady-state phase.

Formally, in terms of Section 2, assume that the update step includes a ℓ_p projection Γ to a compact set with a smooth boundary: $\theta_{m+1} = \Gamma(\theta_m + \eta_m \nabla_\theta \hat{J}_\alpha)$; and that the learning rate η_m satisfies $\sum_{m=0}^{\infty} \eta_m = \infty$, $\sum_{m=0}^{\infty} \eta_m^2 < \infty$ and $\sum_{m=0}^{\infty} \eta_m \left| E\left[\nabla_\theta \hat{J}_\alpha\right] - \nabla_\theta J_\alpha \right| < \infty$ w.p. 1. In addition, denote by $\mathcal{K}$ the set of all asymptotically-stable equilibria of the ODE $\dot{\theta} = \Gamma(\nabla_\theta J_\alpha(R;\pi_\theta))$.

Theorem 2 (Convergence of CeSoR)**.** Assume that for any ϕ, the sample distribution D_ϕ of Algorithm 1 has the same support as the original distribution D_{ϕ_0}. Then, under the smoothness assumptions specified in Appendix C.1, and the projection and learning rate assumptions specified above, the sequence of policy parameters $\{\theta_m\}$ generated by Algorithm 1 converges almost surely to $\mathcal{K}$.

Theorem 2 relies on similar assumptions to Tamar et al. [2015b], two of them are of particular interest in our context. First, the rewards are assumed to be continuous. Second, in the gradient estimator, the baseline is assumed to be a consistent estimator of the returns α-quantile. Hence, while CeSoR is compatible with any CVaR-PG method, the current derivation of theoretical convergence guarantees only holds for PG methods with a consistent gradient estimate.

C.1 Gradient Estimation Bias

The gradient estimator of the standard CVaR PG may be inconsistent and unboundedly-biased, unless the return baseline is a consistent estimator of the α-quantile of the returns [Tamar et al., 2015b]. Thus, we rely on the empirical quantile baseline $\hat{q}_\alpha$ used in Equation (3), which is a consistent (though biased) estimator of the true quantile. Given certain smoothness assumptions, Tamar et al. [2015b] bound the resulted bias of the gradient estimator $E\left[\nabla_\theta \hat{J}_\alpha\right] - \nabla_\theta J_\alpha$ (as defined in Equations (2),(3)). Lemma 2 guarantees that under the same assumptions, despite the modified sampling by the CEM, the same bias bounds apply to CeSoR.

We first specify the smoothness assumptions. Note that Tamar et al. [2015b] consider $\nabla_\theta \log f_{s|a}(s|a,\theta)$ in their calculations (or in their notation: $\nabla_\theta \log f_{X|Y}(X|Y,\theta)$). In RL applications, given the action a, the next-state distribution is independent of the policy π_θ, and this gradient vanishes. We accordingly ignore this term in the calculations, which simplifies the assumptions and the analysis. The remaining assumptions mostly consider the smoothness of the rewards, and in particular do not hold in the case of discrete rewards as discussed in Section A.

Assumption 1 (Smoothness assumptions)**.** For any policy π_θ, the return R is a continuous random variable; and $\nabla_\theta q_\alpha(R;\pi_\theta)$, $\nabla_\theta J_\alpha(\pi_\theta)$ and $\nabla_\theta \log \pi_\theta(a)$ (for any a) are well defined and bounded.

Lemma 2 (Gradient estimation bias bound)**.** In Algorithm 1 with a batch size N, consider a certain step $m \geq \rho M$, and assume that the underlying PG follows Equation (3) (or Equation (6)). In addition, assume that for any ϕ, the sample distribution D_ϕ of Algorithm 1 has the same support as the original distribution D_{ϕ_0}. Then, under Assumption 1, $E\left[\nabla_\theta \hat{J}_\alpha\right] - \nabla_\theta J_\alpha = \mathcal{O}(N^{-1/2})$.

Proof. We follow the steps of the proof of Theorem 4 in Tamar et al. [2015b] with the following modifications. First, we take the gradient expectations with respect to the CE sampling distribution D_ϕ rather than the original distribution D_{ϕ_0}. Second, the empirical quantile $\hat{q}_\alpha$ is calculated in Algorithm 1 using a reduced sample size $N_o = \lfloor \nu N \rfloor < N$. Note that the estimator $\hat{q}_\alpha$ relies on samples drawn from D_{ϕ_0}, hence is not otherwise affected by the CEM.

Denote by D_{ϕ_i} the distribution from which was drawn C_i, i.e., $\phi_i = \phi_0$ for $i \leq N_o$ and $\phi_i = \phi$ for $i > N_o$. Since $m \geq \nu N$, according to Line 13 in Algorithm 1 we have $\alpha' = \alpha$. Denoting by q_α the true α-quantile of the returns, we have

$$\nabla_\theta J_\alpha(R;\pi_\theta) = E_{\{\phi_i\}_{i=1}^N}\left[\frac{1}{\alpha N}\sum_{i=1}^{N} w_i \mathbf{1}_{R_i \leq q_\alpha}\left(R_i - q_\alpha\right)\nabla_\theta \log \pi_\theta(\tau_i)\right] \tag{7}$$

We now substitute $w_i = \frac{D_{\phi_0}(C_i)}{D_{\phi_i}(C_i)}$, which is finite due to the assumption that D_{ϕ_i} has the same support as D_{ϕ_0}. Using the notation $E_{\phi_0}\left[\cdot\right] = E_{C\sim D_{\phi_0}, \tau\sim P_C^{\pi_\theta}}\left[\cdot\right]$ and $\pi_\theta(\tau_i) = \Pi_t \pi_\theta(a_{i,t}; s_{i,t})$, we obtain

$$\begin{aligned}
&\left| E_{C_i\sim D_{\phi_i}, \tau_i\sim P_{C_i}^{\pi_\theta}} \left[\nabla_\theta \hat{J}_\alpha(\{\tau_i\}; \pi_\theta) \right] - \nabla_\theta J_\alpha(\pi_\theta) \right| \\
&\leq E_{C_i\sim D_{\phi_i}, \tau_i\sim P_{C_i}^{\pi_\theta}} \left[\frac{1}{\alpha N} \sum_{i=1}^{N} \frac{D_{\phi_0}(C_i)}{D_{\phi_i}(C_i)} \left| \nabla_\theta \log \pi_\theta(\tau_i) \left(\mathbf{1}_{R_i \leq \hat{q}_\alpha} \left(R_i - \hat{q}_\alpha\right) - \mathbf{1}_{R_i \leq q_\alpha} \left(R_i - q_\alpha\right) \right) \right| \right] \\
&= E_{\phi_0} \left[\frac{1}{\alpha N} \sum_{i=1}^{N} \left| \nabla_\theta \log \pi_\theta(\tau_i) \left(\mathbf{1}_{R_i \leq \hat{q}_\alpha} \left(R_i - \hat{q}_\alpha\right) - \mathbf{1}_{R_i \leq q_\alpha} \left(R_i - q_\alpha\right) \right) \right| \right] \\
&= E_{\phi_0} \left[\frac{1}{\alpha N} \sum_{i=1}^{N} \left| \nabla_\theta \log \pi_\theta(\tau_i) \left(\left(\mathbf{1}_{R_i \leq \hat{q}_\alpha} - \mathbf{1}_{R_i \leq q_\alpha} \right) \left(R_i - \hat{q}_\alpha\right) + \mathbf{1}_{R_i \leq q_\alpha} \left(\left(R_i - \hat{q}_\alpha\right) - \left(R_i - q_\alpha\right) \right) \right| \right] \\
&\leq E_{\phi_0} \left[\frac{1}{\alpha N} \sum_{i=1}^{N} \left| \nabla_\theta \log \pi_\theta(\tau_i) (\mathbf{1}_{R_i \leq \hat{q}_\alpha} - \mathbf{1}_{R_i \leq q_\alpha}) \left(R_i - \hat{q}_\alpha\right) \right| \right] \\
&\quad + E_{\phi_0} \left[\frac{1}{\alpha N} \sum_{i=1}^{N} \left| \nabla_\theta \log \pi_\theta(\tau_i) \mathbf{1}_{R_i \leq q_\alpha} \left(q_\alpha - \hat{q}_\alpha\right) \right| \right]
\end{aligned} \tag{8}$$

From this point, the proof is mostly identical to Theorem 4 in Tamar et al. [2015b]. Namely, the first term is $o(N^{-1/2})$ according to Hong and Liu [2009], given Assumption 1; and since $\hat{q}_\alpha$ is estimated using νN samples, we have $|q_\alpha - \hat{q}_\alpha| = \mathcal{O}((\nu N)^{-1/2}) = \mathcal{O}(N^{-1/2})$ in probability (note that ν is constant, e.g., $\nu = 0.2$ or $\nu = 0.5$ in the experiments of Section 5). Together, the whole expression is $\mathcal{O}(N^{-1/2})$ as required.

□

D The Cross Entropy Module: Extended Discussion

The Cross Entropy Method (CEM) with non-stationary score function has a major role in CeSoR. The CEM code is implemented and available as an independent module [Greenberg, 2022]. Below we present an analysis of the CEM empirical results over both a dedicated toy problem (which tests the CEM independently of CeSoR) and as part of CeSoR in the benchmarks of Section 5.

D.1 The CEM Algorithm

For clarity, we first provide the pseudo-code for the general CEM algorithm. This version repeatedly generates samples from the tail of the distribution D_{ϕ_0}. A similar version [de Boer et al., 2005] would stop once $q_\beta\left(\{R(x_i)\}_{i=1}^N\right) \leq q$ (as it means that at least βN samples are already beyond q), and use all the recent samples $R(x_i) \leq q$ to estimate the probability of the "rare event" $R(X) \leq q$.

Note that unlike CeSoR, Algorithm 2 relies on a constant mapping $R(x)$ and a constant target q. Our CEM version in CeSoR, as implemented in our code and presented in Algorithm 1, supports a quantile-target α with respect to a return mapping R that varies dynamically with the learning agent.

Algorithm 2: The Cross Entropy Method for Sampling

```
Input: distribution $D_{\phi_0}$; score function $R$; target level $q$; batch size $N$; update selection rate $\beta$.
$\phi \leftarrow \phi_0$
while true do
    // Sample
    Sample $x \sim D_\phi^N$
    $w_i \leftarrow D_{\phi_0}(x_i)/D_\phi(x_i) \quad (1 \leq i \leq N)$
    Print $x$
    // Update
    $q' \leftarrow \max\left(q,\ q_\beta\left(\{R(x_i)\}_{i=1}^N\right)\right)$
    $\phi \leftarrow \text{argmax}_{\phi'} \sum_{i=1}^N w_i \mathbf{1}_{R(x_i) \leq q'} \log D_{\phi'}(x_i)$
```

D.2 Sample Distribution

The goal of the CEM is to align the sample distribution with the bottom-α percent of the reference distribution. Note that given a parametric family of distributions D_ϕ with a limited expressiveness, a perfect alignment is not always possible. For example, if the CEM controls the mean of an exponential distribution $C \sim Exp(\phi)$, and the returns decrease with c, then the lower quantiles of the returns correspond to $C \geq q_\alpha(C)$. However, no value of ϕ could eliminate the lower values $C \in [0, q_\alpha]$ – but could merely assign more probability density to higher values. Even when the family of distributions is expressive enough, the CEM has to learn the desired sample distribution without any prior knowledge about the meaning of the parameters that it controls. In particular, it cannot know in advance in which direction each parameter may affect the agent return, what the size of the effect would be, and how it would change during the training.

Formally, the objective of the CEM is often defined as minimization of the KL-divergence between the sample distribution and the desired tail of the reference distribution [Dambreville, 2006]. Indeed, this objective is well-defined even if the expressiveness of D_ϕ does not allow a perfect alignment.

In this section, we focus on the comparison between the mean and the CVaR of the sample distribution and the reference distribution of the returns. Specifically, while both distributions begin with the same mean and CVaR, we hope that the sample mean would align with the reference CVaR as quickly as possible.

First, we consider a toy problem with a static reference distribution and no RL environment. The parametric family of distributions is $C \sim Beta(2\phi, 2 - 2\phi)$ (such that $E[C] = \phi$), and the reference distribution corresponds to $\phi_0 = 0.5$, which results in the uniform distribution $Beta(1, 1) = U(0, 1)$. We are interested in the bottom $\alpha = 10\%$ of the reference distribution, i.e., $U(0, 0.1)$. We run the CEM for $n = 10$ steps with $N = 1000$ samples per step, $\nu = 20\%$ of them are drawn from the original reference distribution, and update ϕ using the mean of the lower $\beta = 50\%$ samples. Note that generally in this work, C is the context or configuration of an environment that produces returns; in this toy example, we do not have an RL environment and we simply define $R(C) = C$.

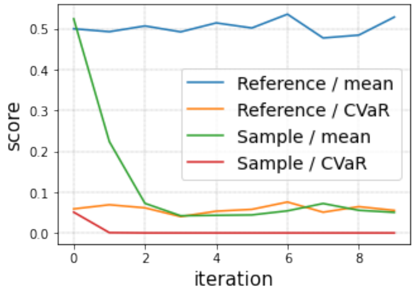
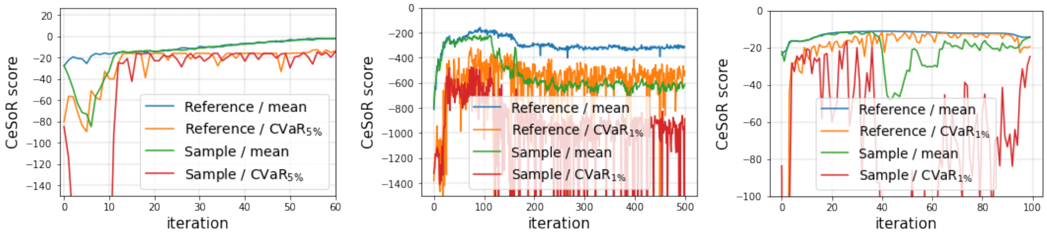


Figure 6: The converges of the CE sample mean to the reference $CVaR_{10\%}$ in the toy $Beta$ distribution problem.

The $CVaR_{10\%}$ of C (or equivalently, the mean of $U(0, 0.1)$) is 0.05. Note that no value of ϕ can yield the distribution $U(0, 0.1)$, as the support of the $Beta$ distribution is always $(0, 1)$. Yet, as shown in Figure 6, the sample mean converges to the reference CVaR within mere 2 iterations, and remains around this level.

Figure 7 presents the same metrics for the experiments described in Section 5. In these cases, the reference returns distribution corresponds to the agent returns under the original environment. Note that this reference returns distribution is dynamic during the training, as it changes with the agent (and in certain benchmarks also with the episode length that increases throughout the training). Yet, in the Driving Game benchmark, for example, we see that the sample mean reasonably aligns with the reference CVaR throughout most of the training, even as both of them vary.



(a) Guarded Maze (first 60 iterations)

(b) Driving Game

(c) Servers Allocation

Figure 7: The mean and CVaR metrics of the CE sample distribution and the reference original distribution throughout the training of CeSoR over different benchmarks.

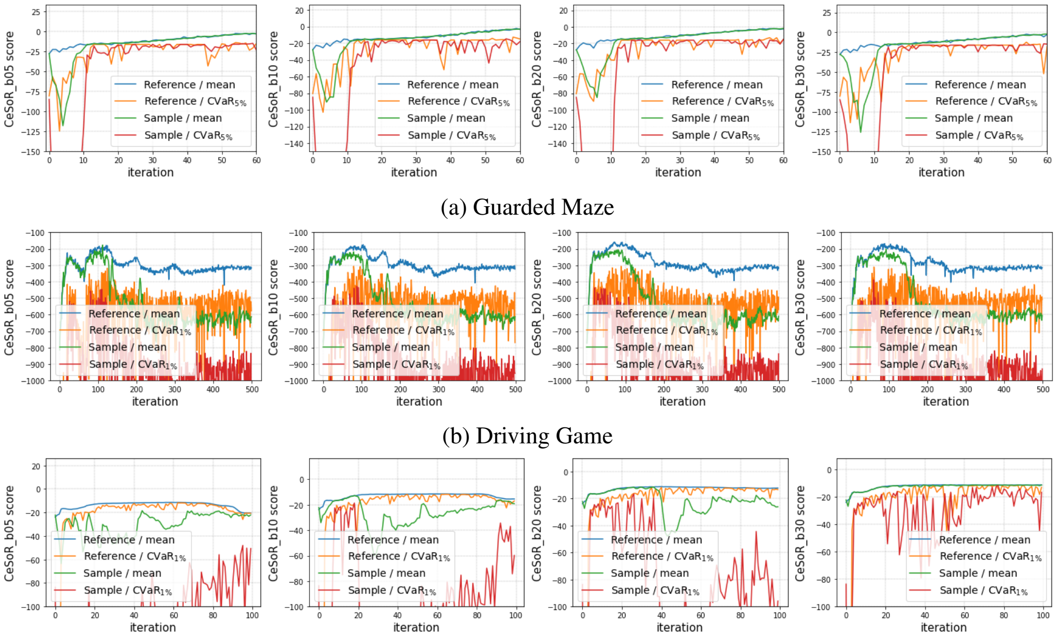
In the Guarded Maze, the sample mean also quickly converges into the reference CVaR. However, once the agent learns to avoid the short path, the CE sampler can no longer control the agent performance at all, and due to the regularizing reference samples, the sample distribution gradually goes back to the original one. This is a valid behavior, as the agent already learned to avoid the risk, and if for some reason it came back to the risky short path, the CE would simply learn again to focus on the risky configurations of the environment.

The Servers Allocation Problem takes the challenge of the CEM to the limit, as the target is $\alpha = 1\%$, the difficulty to the agent arrives in a non-smooth manner as rare and discrete events, and the given family of distributions (Binomial) has limitations in expressing the desired distribution. Specifically, we would like most of the sample episodes to include a peak event, but not more than one; whereas the Binomial distribution is not best-suitable for this. However, even as the CEM struggles to fit the reference $CVaR_{1\%}$ (Figure 7c), CeSoR is still shown to provide beneficial results (Section 5.3, Appendix G). This demonstrates the robustness of CeSoR to limitations and misspecification of the modeled family of distributions.

Sensitivity to β: As discussed in Sections 2 and 4, the smoothness parameter β determines the minimal percent of data samples used for the update step in the CEM. We argue that CeSoR has a low sensitivity to the parameter β.

Intuitively, every iteration of the CEM focuses on the β-tail of the previous iteration (until reaching the α-tail of the reference distribution). Theoretical analysis of the convergence rate is challenging, due to the limited expressiveness of D_ϕ and the non-stationary agent returns; yet, according to the qualitative intuition above, we expect exponential convergence to the tail, which applies even for high values of β. On the other hand, while low values of β may increase the noise in the update step of the CEM, any noisy update could be corrected throughout the training. Note that Algorithm 1 uses the original context-distribution for a certain part of the samples of each batch; this guarantees that any update step is reversible, as CeSoR continues to be exposed to the complete context-space.

Empirically, we repeated the experiments of Section 5 with various values of $\beta \in [0.05, 0.3]$. In the Guarded Maze and the Driving Game, all the values of β resulted in similar test returns; in addition, Figure 8 shows that the CEM successfully aligned the sample mean return with the reference CVaR, independently of β. The Servers Allocation Problem is more challenging for the CEM (as discussed above), making the sampler more sensitive to the parameter β, and in particular leading to a failure for $\beta = 0.3$. However, note that even under such a combination of poor algorithmic choices (Binomial parameterization of D_ϕ and very high β), the failure of the CEM is easy to notice through Figure 8c (as the sample-mean fails to deviate from the reference-mean), and is easy to fix.



(a) Guarded Maze

(b) Driving Game

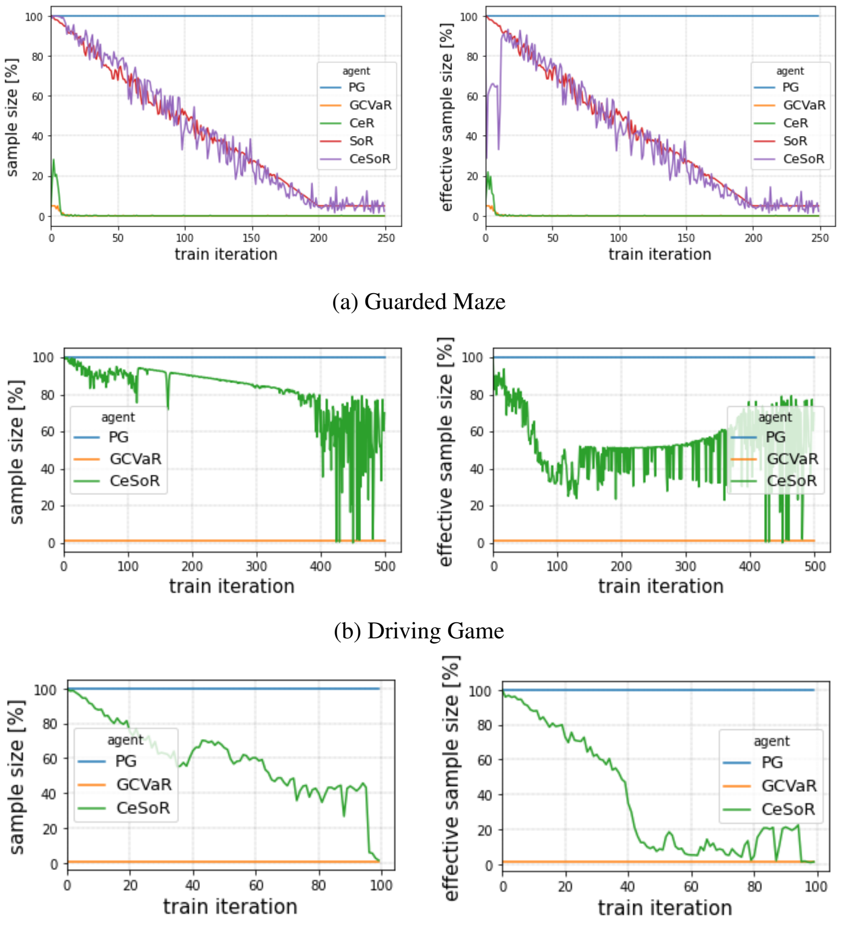
(c) Servers Allocation

Figure 8: Returns statistics of the sample distribution and the reference original distribution, throughout the training of CeSoR over different benchmarks, for different values of $\beta \in [0.05, 0.3]$.

D.3 Sample Efficiency

An important aspect of the CEM is its increase of sample efficiency (Section 3.2). While the results in Section 5 already demonstrate that CeSoR learns better and faster than the standard GCVaR, here we measure the effective sample size directly. While PG always uses the entire batch, and GCVaR always uses at most α of the episodes, Figure 9 shows that CeSoR manages to optimize CVaR$_\alpha$ while using more than α percent of the data. Note that even beyond the risk level scheduling (which ends after $\rho = 80\%$ of the training), the CEM still allows for more than α percent of each batch to be used.

Note that GCVaR effectively uses *less* than α episodes in a batch if multiple episodes $\{\tau_i\}$ satisfy $R(\tau_i) = q_\alpha$ – since the contribution of any such episode to the gradient in Equation (3) is 0. In the extreme case, as discussed in in Section 3.1 and Appendix E.2, all the worst α episodes are identical, and the whole loss gradient is identically 0.



(a) Guarded Maze

(b) Driving Game

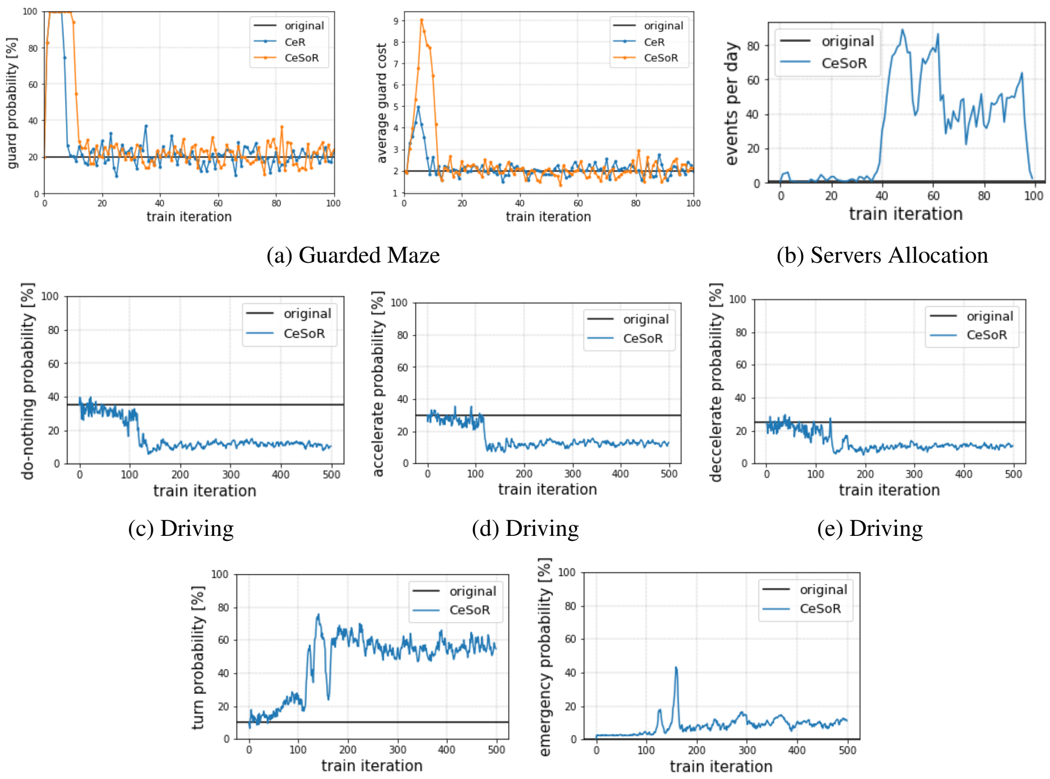
(c) Servers Allocation

Figure 9: Left – sample size: the percent of episode samples (out of $N = 400$ episodes per training iteration) used by the optimizer. Note that only returns $R(\tau_i) < q_\alpha$ are counted (strict inequality), since the contribution of episodes with $R(\tau_i) = q_\alpha$ to the loss is 0 (Equation (3)). Right – *effective* sample size: this takes into account the IS weights: the effective sample size equals the number of equally-weighted independent samples needed to obtain the same estimation variance [Kish, 1965, Leinster, 2014]: $n_{eff} = (\sum_i w_i)^2 / \sum_i w_i^2$. Note that for equal weights, $n_{eff} = n$.

D.4 Risk Characterization

The CEM not only allows CeSoR to sample the most relevant environment conditions for CVaR optimization, but also allows us to characterize the conditions that correspond to the risk level α. This enhances our understanding of the problem and may help us to anticipate poor returns in advance.

Figure 10 presents the evolution of the sample distribution parameters ϕ throughout the CeSoR training process in the various benchmarks. In the Guarded Maze, for example, ϕ goes back to its original values once the agent behavior converges, which teaches us that a risk-averse agent can be entirely insensitive to the environment conditions. In the Driving Game, on the other hand, the agent must still beware a leader that applies many turns and emergency brakes. Furthermore, the CEM provides the connection between the risk level of interest (α) and the corresponding values of ϕ (e.g., how many turns and brakes it takes to bring us to this risk level).



(a) Guarded Maze

(b) Servers Allocation

(c) Driving

(d) Driving

(e) Driving

(f) Driving

(g) Driving

Figure 10: The evolution of the CE distribution parameters ϕ' throughout the training in various benchmarks.

E The Guarded Maze: Extended Discussion

E.1 Implementation Details

In this section we specify the implementation details of the Guarded Maze. The full code is available in the gym environment and the corresponding jupyter notebook.

The Guarded Maze benchmark: The benchmark introduces a maze of size 8×8, with the walls marked in gray in Figure 1d. The target is a 1×1 square marked in green. Every episode, the initial agent location is drawn from a uniform distribution over the lower-left quarter of the maze. Every time step, the agent can walk in one of the directions left, right, up and down, with a step size of 1, and an additive normally-distributed noise with standard deviation of 0.2 in each dimension. That is,

$$s_{t+1} = s_t + a_t + (\epsilon_1, \epsilon_2)^\top$$

where $s_t, a_t \in \mathbb{R}^2$ and $\epsilon_i \sim \mathcal{N}(0, 0.2^2)$ $(i \in \{1, 2\})$. A step that ends in a wall is cancelled, and the agent remains in its place.

Every time-step, the agent observes its location s_t. In practice, we use a soft (continuous) one-hot encoding of the agent location in the maze, calculated as a 2D interpolation between the 4 nearest points of a 8×8 grid, represented as a corresponding 8×8 matrix. That is, if the agent is located between the grid points $(i, j), (i, j + 1), (i + 1, j), (i + 1, j + 1)$, then all the other elements of the matrix are set to 0, and these 4 elements are assigned positive value that are summarized to 1, according to the relative location of the agent between them. Note that the locations of the target and the guarded zone are constant, and are not given as input.

An episode ends either when reaching the target or after 160 time-steps. The rewards are specified in Section 5.1. The return of an episode is the sum of its rewards (i.e., no discount factor). The maze is designed such that the *mean*-optimal strategy is taking the shortest path to the target, where the expected cost of crossing the guarded zone is $E\left[C_1 C_2\right] = \phi_1 \phi_2 = 0.2 \cdot 32 = 6.4$ – smaller than the additional cost of the longer path. The $CVaR_{0.05}$-optimal strategy, however, is to take the longer path, since sometimes short cuts make long delays [Tolkien, 1954].

Algorithms implementation: The training algorithms are specified in Section 5. In the maze benchmark, all of them are applied to a linear model that takes as an input the one-hot encoding described above ($\in \mathbb{R}^{64}$), and is followed by a softmax operator with temperature T. That is, $P(a_j;\theta) = exp(Ty_j)/\sum_{j'} exp(Ty_{j'})$ (where $1 \leq j \leq 4$ and y_j is the corresponding output of the linear model F_θ). We set a constant $T = 1$ over the whole training, and $T = 0$ (i.e., choosing the max-probability action) for validation and test episodes.

The CE module in CeR and CeSoR controls the parameters ϕ of the Bernoulli and the Exponential distributions. Note that the module is aware of the original ("true") values of ϕ, but not of their semantic meaning in the maze (e.g., it is not aware that high values are "bad", or that they only affect the agent through the guarded zone). The sample parameters update using the moments-method is as simple as $\phi \leftarrow (mean(C_1), mean(C_2))$, calculated over the episodes selected by the CE (Line 12 in Algorithm 1).

E.2 Detailed Results

Figure 11 shows the distribution of the trained agent returns over the test episodes in the Guarded Maze (note that the left tail of this distribution is displayed in Figure 1a. Figure 12 shows the mean and CVaR of the training and validation scores throughout the training process. Below we elaborate on the training dynamics in general, and the blindness to success in particular.

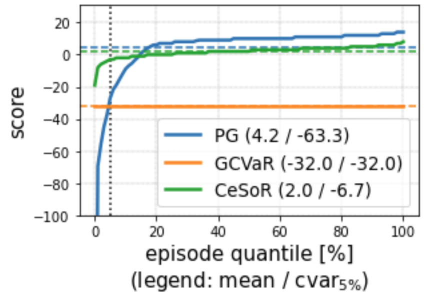


Figure 11: The full distribution of the trained agent returns over the test episodes in the Guarded Maze. Note that Figure 1a displays the left tail of the same distribution.

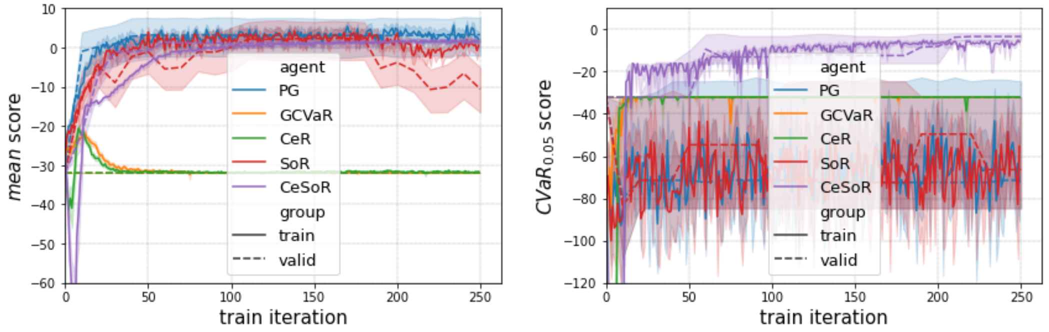
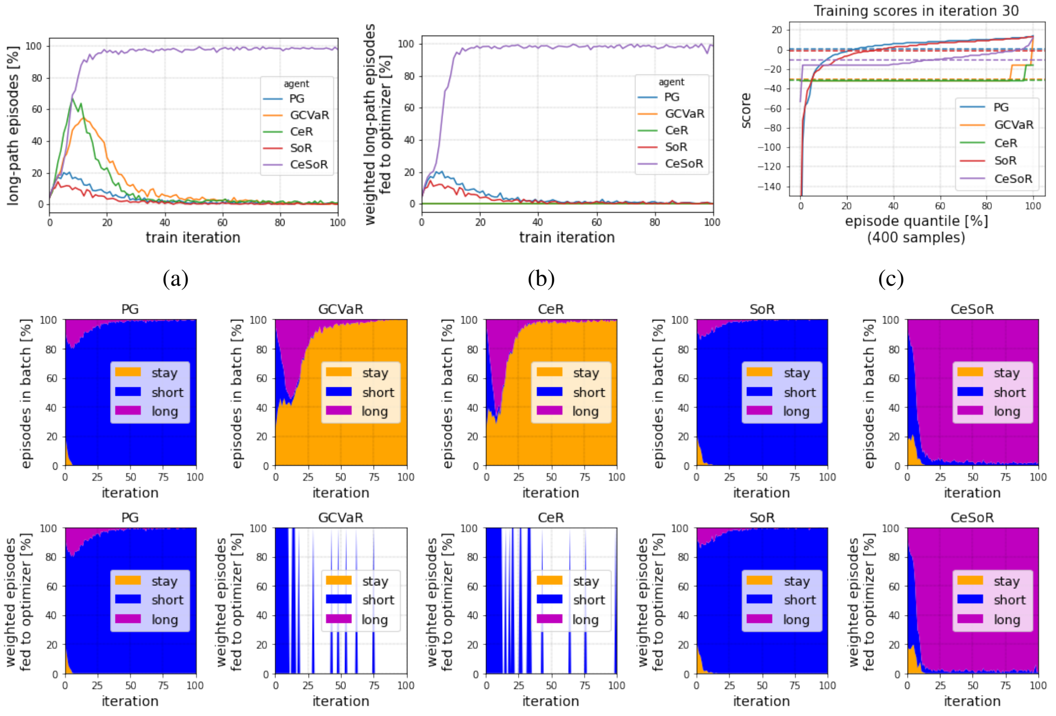


Figure 12: Mean and CVaR scores over the train and validation episodes throughout the Guarded Maze training. The shading corresponds to 95% confidence-intervals, based on bootstrapping over the episode-samples. Note that validation and train policies are not entirely identical, as the former deterministically chooses the action of max-probability (temperature $T = 0$), and the latter operates stochastically ($T = 1$).

Blindness to success: Section D.3 discusses the contribution of the *CE sampling* to the sample efficiency. Here we discuss the contribution of *soft risk level scheduling* to the sample efficiency, and in particular its prevention of *blindness to success*.



(a) (b) (c)

(d)

Figure 13: For the first 100 iterations of the Guarded Maze training, (a) the percent of episodes that reached the target through the long path; (b) the total weight of such long-path episodes that were fed to the optimizer (out of the total weight of episodes fed to the optimizer); (c) the returns distribution over the 30th training batch; and (d) percent of episodes (top) and total weight (bottom) for all 3 agent strategies (not only long path as in (a),(b)).

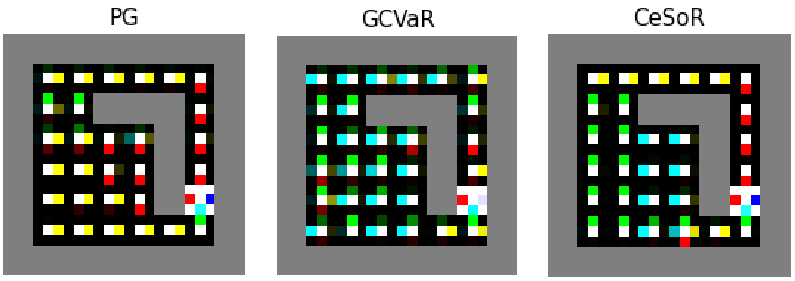
As displayed in Figure 13a, for all the agents in the beginning of the optimization process, around 10% of the episodes in every batch reach the target through the long path. At the same time, around 70% of the episodes reach the target through the short (and risky) path. As a risk-averse algorithm, GCVaR learns to avoid the short path, and the ratio of the long-path episodes increases accordingly – reaching up to 50% around the 15th batch (recall that in training episodes the actions are selected randomly according to the policy softmax output with temperature 1, which allows the agent to randomly reach the target). Nonetheless, as shown in Figure 13b, in *all* of the train iterations, *none* of the long-path episodes belong to the bottom $\alpha = 5\%$ episodes (which are fed to the optimizer), hence GCVaR never learns to prefer the long-path. This demonstrates the blindness of GCVaR to the successful long path.

In fact, after around 10 training iterations of GCVaR, all the bottom $\alpha = 5\%$ episodes in most batches already follow the stay-strategy (i.e., do not reach the target, nor take the guarded-zone risk), and achieve a constant return of -32 (Figure 13c). Note that according to Equation (3), this means that the loss gradient is identically 0. As shown in Figure 9a, the used sample size of GCVaR is indeed 0 after the 10th iteration, the effective sample efficiency is 0, and most of the changes in the agent from this point are attributed to the remaining Adam gradient momentum.

The soft risk level scheduling eliminates the blindness to success, and allows the optimizer to observe the long-path episodes (SoR in Figure 13b). However, at the same time, it reduces the risk-aversion of the agent, and the long path is no longer preferred over the short path. When the risk level reduces sufficiently, the agent may re-learn to avoid the short path, but the long path is no longer sampled at all and cannot be learned.

Only CeSoR manages both to observe the long-path episodes (thanks to soft risk level scheduling) *and* to prefer them over the short path (thanks to the risk-aversion induced by the CEM).

Examples and visualization: Figure 14 visualizes the policies learned by PG, GCVaR and CeSoR. While the policies are defined over all the continuous state space, the visualization is restricted to a discrete grid. Note that CeSoR and GCVaR behave similarly in the lower-left part of the maze, corresponding to guarded-zone avoidance; however, since GCVaR never observed the long path and learned its benefits, it fails to learn the CVaR-optimal strategy in the upper part of the maze.



(a) (b) (c)

Figure 14: The policies learned by PG, GCVaR and CeSoR, visualized over a discrete grid within the continuous state space of the Guarded Maze. The colors brightness around each point in the grid corresponds to the probabilities assigned to the actions by the policy given this point.

Figure 15 shows a sample of test episodes for each of the trained agents. Due to the reduced risk-aversion of SoR (as discussed above), its best validation CVaR score was obtained early in the training, which may explain its non-smooth behavior in Figure 15.

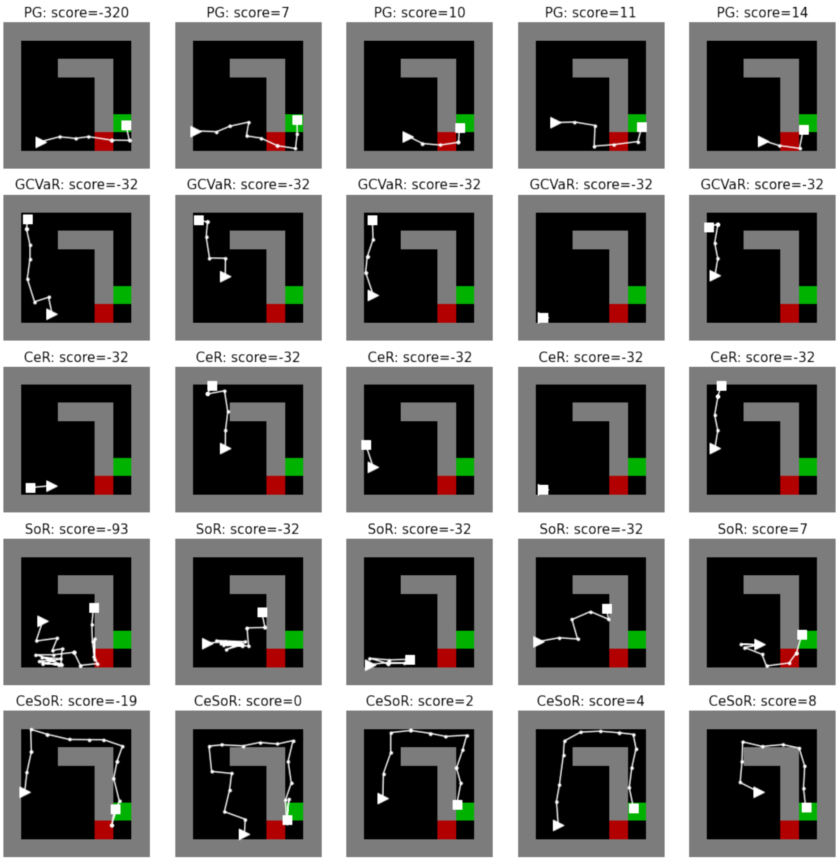


Figure 15: A sample of test episodes for each of the trained agents in the Guarded Maze.

F The Driving Game: Extended Discussion

F.1 Implementation Details

In this section we specify the implementation details of the Driving Game. The full code is available in the gym environment and the corresponding jupyter notebook. Note that the leader behavior generation mechanism and the policy architecture are already specified in Section 5.

Observation space: the policy receives the following variables as inputs: relative position dx, dy, relative on-track velocity dvx, agent acceleration ax and agent direction θ.

Action space: the possible agent actions are (1) keep speed and steer; (2) accelerate; (3) decelerate; (4) steer left; (5) steer right. The acceleration and deceleration magnitudes ($+4m/s^2$, $-6m/s^2$) were determined according to the typical acceleration value described in Singh et al. [2018].

Rewards: we use the rewards defined in Singh et al. [2018], with the parameters $r_1 = 0.5, r_2 = 0.05, r_3 = 0.1, r_4 = 0.5, r_5 = 1, r_6 = 0.5$. These parameters determine the scale of the 6 additive rewards of Singh et al. [2018], which correspond to staying behind the leader, staying close to the leader, keeping similar speed to the leader, keeping smooth agent acceleration, staying in the same lane as the leader, and staying on-road, respectively. We also add a new additive reward of size 5 for any time-step with overlap between the agent and leader cars, meant to penalize collisions – which are not explicitly expressed in the original rewards.

F.2 Detailed Results

Figures 16-18 present a detailed analysis of the results of the Driving Game experiments.

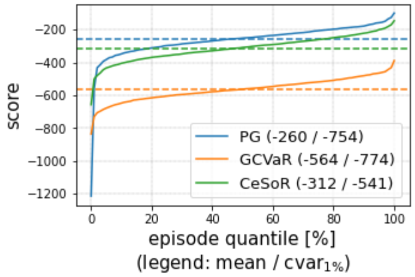
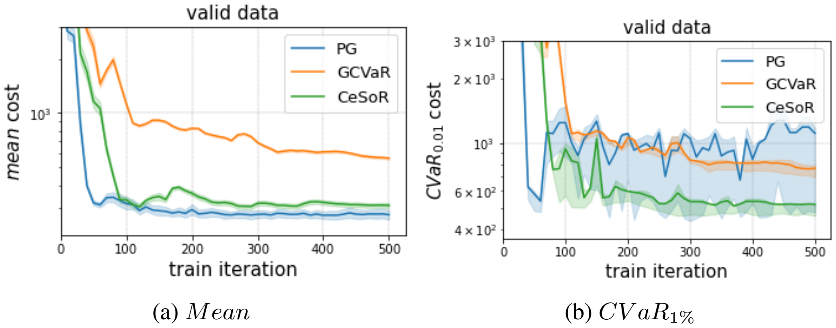


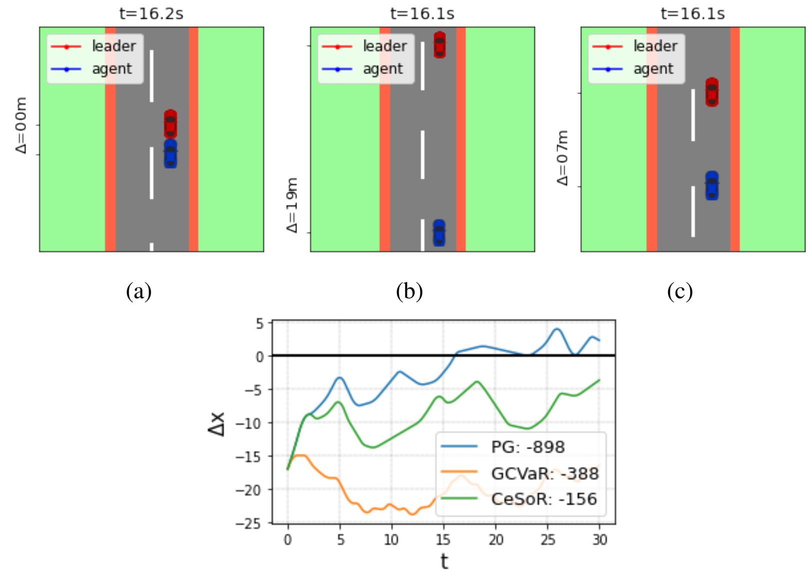
Figure 16: The full distribution of the trained agent returns over the test episodes in the Driving Game. Note that Figure 1b displays the left tail of the same distribution.



(a) $Mean$

(b) $CVaR_{1\%}$

Figure 17: Mean and CVaR scores over the validation episodes throughout the Driving Game training. The shading corresponds to 95% confidence-intervals, based on bootstrapping over the episode-samples.



(a) (b) (c)

(d)

Figure 18: (a-c) A sample frame in a test episode in the Driving Game. All the agents deal with the same situation (the same sequence of leader actions, which happened to include a sequence of decelerations). While PG collides with the leader, CeSoR keeps a safe margin – without losing as much distance as GCVaR. Note that Figure 1e effectively displays these 3 frames together. (d) The agent-leader distance evolution in the whole episode, and the final episode score of each agent.

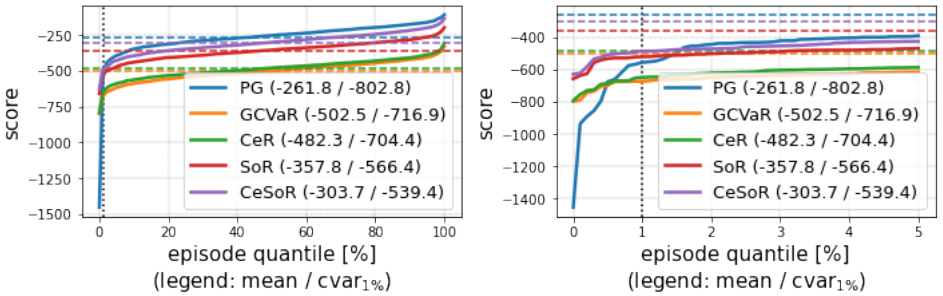


Figure 19: Additional ablation tests for the Driving Game: the full returns distributions (left) and zoom-in to their tails (right). Note that we reran the experiment for the ablation test, resulting in slightly different returns than Figure 1. Both CeR and SoR lose to CeSoR in terms of CVaR and mean, indicating the necessity of both soft risk and CE-sampler in CeSoR.

G The Computational Resource Allocation Problem: Extended Discussion

G.1 Implementation Details

In this section we specify the implementation details of the Resource Allocation Problem presented in Section 5.3. The full code is available in the gym environment and the corresponding jupyter notebook.

The benchmark simulates one-hour episodes, where user-requests arrive randomly and the agent is responsible to allocate sufficiently many servers to handle them. Once a request is attended, its service time is distributed exponentially with an average of 1 second. Every second t, the number of arrivals is distributed $\sim Exp(\lambda_t)$, where the arrival rate λ_t is itself an exponential moving average (EMA) of the (unknown) users interest r_t, with a typical decay of 5 minutes (i.e., $\lambda_t = \frac{299}{5\cdot60}\lambda_{t-1} + \frac{1}{5\cdot60}r_t$). $r_t = 3$ is usually constant, but an unpredictable event causes a peak load every second with probability $\phi_0 = \frac{1}{3\cdot24\cdot3600}$, i.e., every 3 days (or 72 episodes) on average. In case of a peak load we set the momentary user interest to $r_t = 3 \cdot 300$, which means that the arrival rate doubles immediately to $\lambda_t = \frac{299}{300}\lambda_{t-1} + \frac{1}{300}r_t = \frac{299}{300}3 + \frac{1}{300}3 \cdot 300 \approx 6$, and then starts decreasing exponentially back to 3, with a typical decay of 5 minutes.

Every minute, the agent observes the number of active servers $3 \leq n^s \leq 10$ (initialized every episode to $n^s = 4$) and the number of pending user-requests in the system, and may choose to add or remove one server (or to keep the number of servers as before). Uploading a new server takes a 2-minute delay before the server is ready to handle requests. Removing a busy server takes effect once the server ends its current task. Note that the servers form an ordered list, and only the last server in the list can be directly removed. This constraint has little significance, since (1) the queue of pending requests is a global FIFO queue (i.e., the assignment only happens when a server becomes available – there is no separate queue per server); (2) the requests serving time is exponentially distributed, i.e., the remaining time of the current task is independent of the task history and thus is identical for all the busy servers at any point of time.

Denoting by tts_i the Time-To-Service (TTS) latency of a request, the agent return is

$$R = -\text{user cost} - \text{servers cost} = -\sum_{i \in \text{requests}} tts_i - 2\sum_{t=1}^{3600} n_t^s.$$

Once a request is assigned to a server, its serving time $\sim Exp(1)$ is independent of the agent decisions. Thus, to simplify computations and to reduce the noise, we measure the TTS of a request only as the waiting time between arrival and beginning of serving.

We set a target risk level of $\alpha = 0.01$, and train each agent for $n = 100$ steps. During the training, we gradually increase the episodes length L from 15 to 60 seconds. The CEM controls the peak events frequency ϕ, or equivalently, the number of peaks per episode (which is distributed $\sim Binom(\phi, L)$). The update function of ϕ is simply the (weighted) average number of peaks per selected episode, divided by the episode length. $\nu = 50\%$ of the episodes per batch are drawn from the original distribution D_{ϕ_0}.

Note that at times of no peak-loads, the arrival rate is $\lambda = 3$ and the service rate equals the number of servers n^s (since the service takes 1 second on average). Thus, in terms of queueing theory, any number of servers $n^s \geq 4$ guarantees that the expected number of requests in the system is $E\left[n^r\right] = 3/(n^s - 3) \leq 3$. In particular, this means that the policy learned by PG (see Section 5.3) chooses the minimal number of servers $n^s = 4$ that can handle no-peak demand, and adds resources only when required.

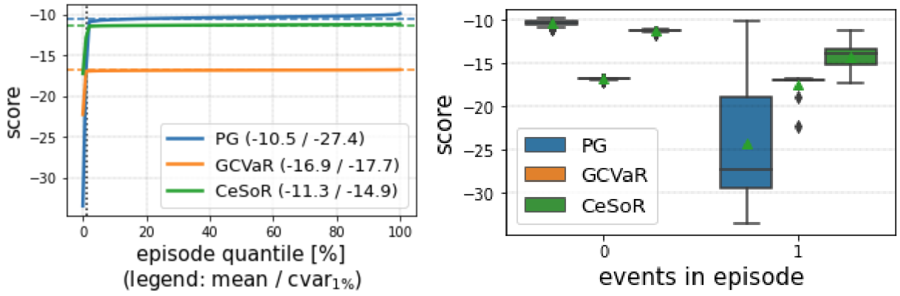
The agent policy receives a 9-dimensional vector as an input. The first 8 elements correspond to a one-hot encoding of the current number of paid servers $3 \leq n^s \leq 10$ (including new servers that are not finished uploading yet). The last element corresponds to the current number of pending user requests in the queue, divided by $10r = 30$ (the average number of arriving requests in 10 seconds of no peak-load).

G.2 Detailed Results

Figure 1c summarizes the test scores of the agents, where CeSoR presents a reduction of 44% and 17% in the CVaR cost in comparison to PG and GCVaR, respectively. In addition, its average cost is only 7% higher than PG, and 33% lower than GCVaR. That is, CeSoR significantly improves the CVaR return without as a large compromise to the mean as in GCVaR. CeSoR also outperforms GCVaR in episodes both with and without peak events, as shown in Figure 20b below. As demonstrated in Figure 1f and summarized in Figure 22, PG and CeSoR learned to allocate a default of 4 and 5 servers, respectively, and to react to peak loads as needed; whereas GCVaR simply allocates 8 servers at all times.

Note that the CE task – sampling the bottom $\alpha = 1\%$ – is particularly challenging in this problem, due to the combination of very rare peak events and limited expressiveness of the Binomial distributions family. In particular, this family cannot guarantee the existence of a peak in a simulated episode without simulating *multiple* peaks per episode (i.e., $P_{\phi^*}^{\pi_\theta} \neq P_{\phi_0,\alpha}^{\pi_\theta}$ in terms of Section 3.2). Yet, CeSoR is demonstrated robust to the poor parameterization selection of D_ϕ, as it presents a reasonable sampling (see Appendix D.2) and improves the returns CVaR.

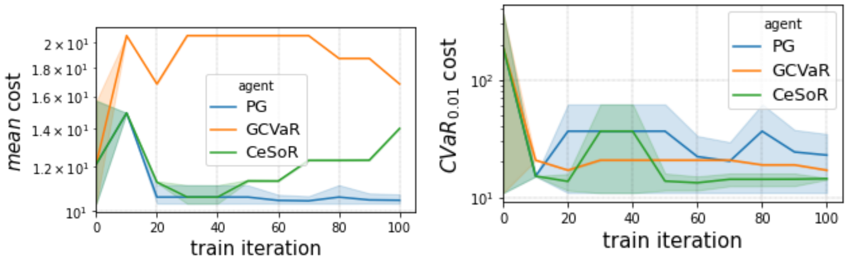
Figures 20-23 present a detailed analysis of the results.



(a)

(b)

Figure 20: (a) The full distribution of the trained agent returns over the test episodes in the Servers Allocation Problem. Note that Figure 1c displays the left tail of the same distribution. (b) A box-plot of the returns distribution for test episodes – separately for episodes with and without a peak-overloading event. CeSoR achieves the best scores in episodes with peak events.



(a) $Mean$

(b) $CVaR_{1\%}$

Figure 21: Mean and CVaR scores over the validation episodes throughout the Servers Allocation Problem training. The shading corresponds to 95% confidence-intervals, based on bootstrapping over the episode-samples.

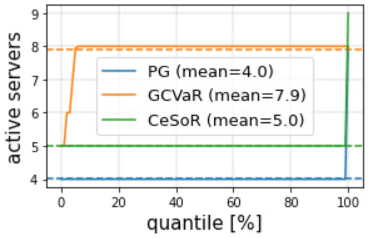
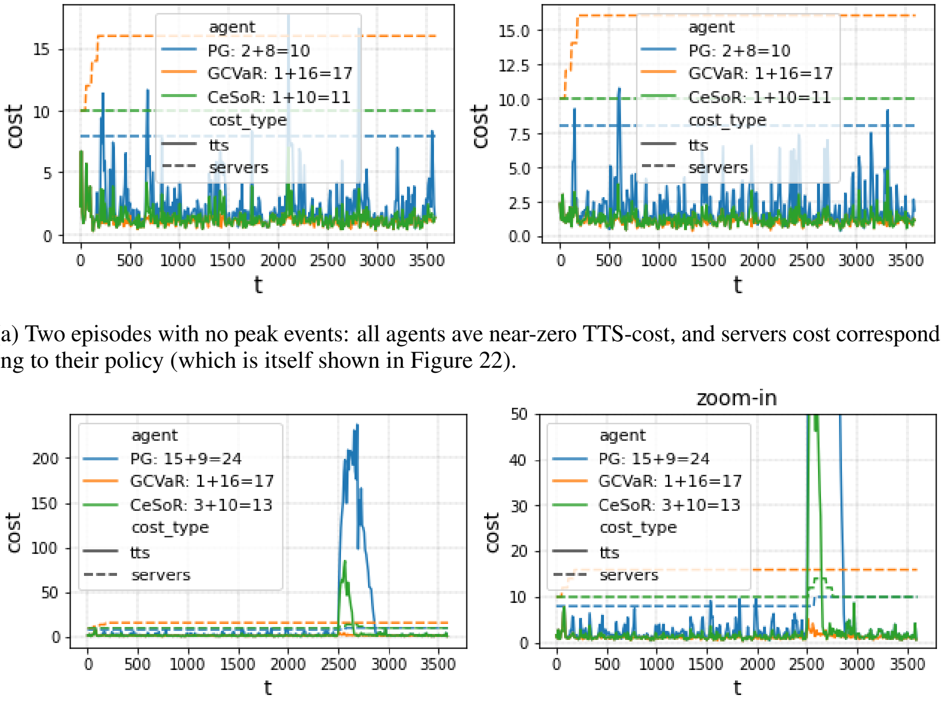


Figure 22: The distribution of the number of servers allocated by each agent, over all the time-steps in all the test episodes. GCVaR allocates 8 servers in advance, whereas PG and CeSoR typically allocate 4 and 5 servers, respectively, and add servers as needed in case of overloading.



(a) Two episodes with no peak events: all agents ave near-zero TTS-cost, and servers cost corresponding to their policy (which is itself shown in Figure 22).

(b) An episode with a peak event (right: zoom in around the event). This figure presents the same episode displayed in Figure 1f, but normalizes the TTS and the servers allocation to the same units of cost, as defined by the benchmark. Notice that both PG and CeSoR react to the event with allocation of additional servers.

Figure 23: A sample of test episodes in the Servers Allocation Problem. The legends specify the TTS-cost, the servers-cost and the total cost.

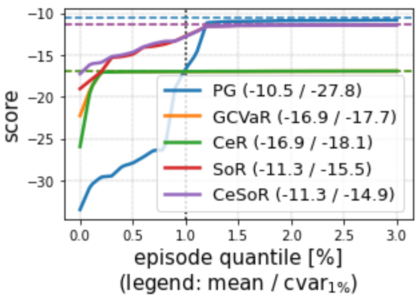


Figure 24: Additional ablation tests for the Servers Allocation Problem. Note that we reran the experiment for the ablation test, resulting in slightly different returns than Figure 1. Both CeR and SoR lose to CeSoR in terms of CVaR and mean, indicating the necessity of both soft risk and CE-sampler in CeSoR.

H Distributional Reinforcement Learning for CVaR Optimization

Many RL algorithms aim to learn the value $Q(s, a)$ of a state-action pair, representing the expected return from choosing action a at state s. Then, given a state and a finite set of action candidates, the agent can choose the action with the highest value. In Distributional Reinforcement Learning (DRL), not only the expected return is learned, but rather the whole return distribution – conditioned on s, a and the current policy. While standard DRL algorithms [Bellemare et al., 2017, Dabney et al., 2018b] still optimize the expected return and thus are risk-neutral, the learning of the whole return distribution encourages risk-averse variants as well [Dabney et al., 2018a].

A naive risk-averse DRL agent may simply use the learned return distribution to choose the action with the highest risk measure (e.g., CVaR) over the returns. However, notice that the return distribution is conditioned on the policy. Hence, similarly to other RL methods, the learned values become incorrect once we change the policy: the CVaR of the current action does not take into account the change in the next action. Thus, this naive approach would not truly optimize the CVaR.

Instead, a risk-averse DRL agent can train using a risk-averse actor, such that the learned distribution is consistent with the risk-averse policy. This approach is valid and was indeed used by Dabney et al. [2018a]. However, it suffers from similar limitations as CVaR-PG. Regarding sample-efficiency, CVaR-DRL considers only the bottom quantiles of the distribution, whose corresponding loss function assigns very low weights to all the returns except for the lowest ones, reducing the effective sample size. In particular, since there is no separation between low returns and high-risk environment conditions, still only a small portion of the data corresponds to high-risk, and it remains challenging to learn how to act under such conditions. Regarding blindness to success, CVaR-DRL is still prone to miss beneficial strategies: it still directs the actor policy according to the lowest returns rather than the hardest conditions, and learns the distribution with respect to that policy.

We implemented the methods mentioned above for the Guarded Maze benchmark, on top of the QR-DQN [Dabney et al., 2018b] implementation of Stable-Baselines [Raffin et al., 2021]. As shown in Table 1, none of the DRL variants improved the CVaR return even in comparison to the baseline CVaR-PG (GCVaR): the standard risk-neutral QR-DQN obtained similar returns to the risk-neutral PG; the naive DRL approach resulted in a noisy and seemingly-meaningless policy, obtaining worse returns than GCVaR; and the valid CVaR-DRL obtained identical returns to GCVaR.

These results support the discussion above, indicating that blindness to success and sample-inefficiency are general limitations in risk-averse RL, and in particular apply to DRL in addition to PG. We hope that our work will pave the way for other efficient risk-averse RL methods, beyond the scope of PG algorithms.

Table 1: A comparison of CeSoR test returns to both PG and Distributional RL methods, over the Guarded Maze benchmark. The first two methods are risk-neutral.

Algorithm	**Mean**	**CVaR$_{0.05}$**
PG	**4**	-63
QR-DQN	**3**	-73
GCVaR	-32	-32
CVaR-QR-DQN (only inference)	-32	-39
CVaR-QR-DQN (training+inference)	-32	-32
CeSoR	2	**-7**