Mean-L^p Risk-Constrained Reinforcement Learning: Primal-Dual Policy Gradient and Augmented MDP Approaches

Anonymous Author(s)
Affiliation
Address
email

Abstract

Convex risk measures allow decision-makers to account for uncertainty beyond standard expectations, and have become essential in safety-critical domains. One widely used example is the Conditional Value-at-Risk (CVaR), a coherent risk metric that targets tail outcomes. In this paper, we consider a more general family of risk measures, the *mean-L^p risk* for $p \geq 1$, defined as the L^p-norm of a cost distribution; this family includes CVaR as an extreme case (as $p \to \infty$). We formulate a reinforcement learning problem in which an agent seeks to maximize reward subject to a mean-L^p risk constraint on its cumulative cost. This problem is challenging due to the nested, non-Lipschitz structure of the L^p risk measure, which hinders the use of standard policy optimization or dynamic programming techniques. To address this, we propose two complementary solution approaches: (1) a **primal-dual policy gradient algorithm** that relaxes the risk constraint via a Lagrange multiplier, and (2) a **model-based dynamic programming method** that enforces the constraint by augmenting the state space with a cost budget. We prove that the policy-gradient approach converges to an ϵ-optimal safe policy with $\tilde{O}(1/\epsilon^2)$ samples, matching the best-known rate for simpler (risk-neutral or linear-constraint) cases. Meanwhile, the augmented MDP method computes a policy that never violates the cost limit and is nearly optimal for large p. Our results provide the first general-purpose algorithms for L^p-risk-constrained RL, generalizing prior approaches that were limited to CVaR or variance-based risk. We validate our theoretical results through experiments in a gridworld environment, demonstrating that both algorithms successfully learn policies that respect the risk constraint and adjust conservativeness as the risk sensitivity parameter p varies. The code is available at `https://anonymous.4open.science/r/Lp-Risk-Constrained-Reinforcement-Learning-11FD/README.md`

1 Introduction

In many stochastic decision problems, it is not sufficient to optimize only the expected outcome; one must also account for *risk* or variability in the outcomes. *Risk-averse optimization* (also known as mean-risk optimization) addresses this need by incorporating a risk measure into the objective function [1]. Convex risk measures, in particular, satisfy desirable axioms for rational risk assessment [2] and have become standard tools in fields like finance, energy, and supply chain management. One well-known example is the Conditional Value-at-Risk (CVaR) [3], which quantifies the expected loss in the worst α-fraction of scenarios and is celebrated for its coherence and tractable optimization properties. Another important class is the *mean-upper- semideviation* risk measure of order $p \geq 1$ [4], which captures higher-moment risk by penalizing the higher-end deviations of losses. This L_p-type risk measure generalizes simpler cases: for instance, $p = 1$ recovers the mean-absolute deviation,

Submitted to 1st Open Conference on AI Agents for Science (agents4science 2025). Do not distribute.

$p = 2$ yields a mean-semivariance metric, and as $p \to \infty$ the measure increasingly emphasizes worst-case outcomes (bridging toward a max-loss criterion). By adjusting the order p, one can flexibly model different degrees of risk sensitivity beyond what CVaR (focused on a fixed tail percentile) offers.

Despite their appeal, general convex risk measures are often much harder to optimize than traditional risk-neutral expectations. The CVaR at a given confidence level α can be optimized relatively efficiently by introducing an auxiliary variable and linearizing the tail loss function [3], or via distributionally robust formulations that turn CVaR into a linear program [1]. In contrast, the mean-L_p risk objective does not admit such a straightforward transformation when $p > 1$. In fact, the mean-L_p risk of a decision $x \in X$ can be written in nested form as

$$\rho_p[x] \;=\; \mathbb{E}[Z_x] \;+\; c\Big(\mathbb{E}[\,(Z_x - \mathbb{E}[Z_x])_+^p\,]\Big)^{1/p},$$

where $Z_x = F(x, \xi)$ is a random cost outcome, $(\cdot)_+ = \max\{\cdot, 0\}$ denotes the positive part, and $c > 0$ is a given risk-aversion weight. This formulation involves a composition of expectation and a power function. Crucially, for $p > 1$ the outer mapping $u \mapsto u^{1/p}$ is *concave* and not globally Lipschitz continuous on $(0, \infty)$, which means standard stochastic gradient methods cannot be directly applied or would suffer poor convergence. Indeed, if one naively treats the above as a two-level nested expectation problem, existing single-timescale stochastic approximation techniques [5] yield a convergence rate on the order of $O(1/\epsilon^4)$ in the accuracy ϵ (even under smoothness assumptions), far worse than the $O(1/\epsilon^2)$ optimal rate for simpler convex objectives. The difficulty stems from the non-convex (though quasi- convex) nesting and the "blow-up" of subgradients caused by the $u^{1/p}$ term near $u = 0$ – informally, the problem is neither smooth nor Lipschitz in the usual sense, despite the overall risk measure $\rho_p[\cdot]$ being convex in x.

To overcome these challenges, we seek principled algorithmic solutions for general L_p risk minimization. It is closely related to a distributionally robust optimization (DRO) formulation: as shown by Shapiro et al. [1, Section 6], the objective $\rho_p[x]$ can be interpreted as the worst-case expected cost under all probability distributions that lie within an L_q-neighborhood of the nominal distribution (with $1/p + 1/q = 1$). This DRO perspective underscores the importance of this risk criterion, but also highlights its computational complexity: unlike the α-CVaR case (which corresponds to an ℓ_1 ambiguity set and is linear), the L_p-ball ambiguity set for $p > 1$ yields a hard nonlinear optimization problem.

Several recent works have studied special cases of related nested optimization problems. For $p = 1$, the risk measure $\rho_1[x] = \mathbb{E}[Z_x] + c\,\mathbb{E}[|Z_x - \mathbb{E}Z_x|]$ is essentially a two-level expectation (a convex composite), which can be solved by advanced stochastic approximation methods at the optimal $O(1/\epsilon^2)$ sample complexity [6]. For general multi-level stochastic programs, Ghadimi et al. [5] proposed a single-timescale stochastic mirror descent approach; however, as noted above, its performance deteriorates on problems like ρ_p due to the non-Lipschitz, concave outer layer. Ruszczyński [7] studied a related class of nonconvex risk nested problems and developed a specialized subgradient method, though without complexity guarantees. Overall, there remains a gap in the literature for efficiently solving the mean-L_p risk minimization problem for $p > 1$ with provable guarantees. This challenge is also evident in safe reinforcement learning, where risk constraints beyond the expectation (or simple proxies like CVaR) have remained difficult to optimize reliably.

In this work, we bridge this gap by presenting the first efficient solution methods for reinforcement learning with a general L^p risk constraint ($p > 1$). Our contributions can be summarized as follows:

1. **Primal-Dual Policy Gradient Algorithm:** We develop a Lagrangian-based policy optimization method (Algorithm 1) that provably converges to an optimal policy under convexity assumptions. By performing simultaneous gradient updates on the policy parameters and a dual variable, our approach achieves an $\tilde{O}(1/\epsilon^2)$ sample complexity to reach an ϵ-optimal, ϵ-feasible solution. Notably, this is the first algorithm with global convergence guarantees for RL under a nonlinear L^p risk constraint.

2. **Augmented State Dynamic Programming:** We propose a model-based planning algorithm (Algorithm 2) that exactly enforces the risk constraint by augmenting the MDP state with the remaining cost budget. Solving this augmented MDP via value iteration yields a policy that never violates the cost limit (satisfying a strict ρ_∞ criterion). We show that this policy is

nearly optimal for the L^p-constrained problem (especially for large p), making Algorithm 2 a reliable baseline for verifying the performance of the policy gradient method.

3. **Broader Implications:** Our framework handles any risk order $p \geq 1$, significantly generalizing prior risk-averse RL methods that focused on variance or CVaR-based criteria. By interpolating between average-case and worst-case extremes, the L^p family enables flexible risk-sensitive policy design to suit different applications. We validate our theoretical results through experiments in a gridworld environment, demonstrating that both algorithms successfully learn policies that respect the risk constraint and adjust conservativeness as the risk sensitivity parameter p varies.

2 Method

2.1 Problem Formulation and Risk Measure

We consider a risk-constrained Markov Decision Process (MDP) defined by $(\mathcal{S}, \mathcal{A}, P, r, c, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, $P(s'|s,a)$ is the transition probability, $r(s,a)$ is the reward, $c(s,a)$ is the cost (encapsulating negative "safety" reward), and $0 < \gamma < 1$ is a discount factor. Let π_θ denote a policy with parameters θ. The agent's performance is measured by the expected return $J_R(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, while safety is quantified by a risk measure applied to the cumulative cost. Specifically, define the random cumulative cost $J_C(\pi) = \sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)$ under policy π. We impose a general L^p risk constraint on $J_C(\pi)$, as introduced by [8]. This L^p-risk measure $\rho_p(J_C(\pi))$ is defined as the L^p-norm of the cost distribution:

$$\rho_p(J_C(\pi)) = \left(\mathbb{E}_\pi\left[J_C(\pi)^p\right]\right)^{1/p} \tag{1}$$

for some $p \geq 1$. This formulation recovers standard criteria as special cases: $p = 1$ gives the conventional expected cost constraint (risk-neutral CMDP), while $p \to \infty$ yields an almost-sure (worst-case) cost constraint. The agent's objective is to maximize reward subject to an L^p-risk safety constraint:

$$\begin{aligned} \max_\pi \quad & J_R(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t)\right] \\ \text{s.t.} \quad & \rho_p(J_C(\pi)) = \left(\mathbb{E}_\pi\left[J_C(\pi)^p\right]\right)^{1/p} \leq \beta \end{aligned} \tag{2}$$

where β is a prescribed risk limit. This formulation generalizes prior risk-constrained RL settings (e.g. using CVaR_α as the risk measure [8]) to a broad class of tail-sensitive criteria. The L^p constraint penalizes variability in the cost: higher p emphasize worst- case outcomes more strongly. We focus on the discounted infinite-horizon case with a finite episodic cutoff at horizon H for ease of analysis; in practice one often lets $H \to \infty$ (as our theoretical guarantees hold in the limit).

In a conventional constrained MDP (CMDP) with an expected cost constraint ($p = 1$), standard Lagrange relaxation techniques can be used to solve for an optimal policy [9, 10, 11]. Our setting is more challenging because $\rho_p(J_C)$ is a nonlinear function of the policy. Nonetheless, we can still leverage a primal-dual approach to handle the constraint.

2.2 Policy Gradient with Lagrangian Relaxation

Our first approach directly optimizes the constrained objective by introducing a Lagrange multiplier for the risk constraint. We form the Lagrangian function for policy π_θ with dual variable $\lambda \geq 0$:

$$\mathcal{L}(\theta, \lambda) = J_R(\pi_\theta) - \lambda\left(\rho_p(J_C(\pi_\theta)) - \beta\right) \tag{3}$$

which penalizes constraint violations when $\rho_p(J_C) > \beta$. The constrained RL problem can then be solved via a saddle-point optimization: maximize $\mathcal{L}$ over policy parameters θ while minimizing over λ (dual ascent). Intuitively, the Lagrange multiplier λ adaptively adjusts the trade-off between reward and risk: if the policy violates the risk limit, λ increases to penalize cost more heavily; if the policy is too conservative (risk well below β), λ may decrease, allowing more reward-seeking behavior.

We adopt an iterative primal-dual policy gradient algorithm (Algorithm 1) to solve $\min_{\lambda \geq 0} \max_\theta \mathcal{L}(\theta, \lambda)$. At each iteration, we evaluate the policy (by simulation or rollout) to estimate

both $J_R(\pi_\theta)$ and the risk measure $\rho_p(J_C(\pi_\theta))$. Notably, $\rho_p(J_C)$ is a nonlinear function of the policy; in practice we approximate its gradient via sampling. For instance, one can use policy gradient for risk measures: Tamar et al. [12] developed gradient estimators for coherent risk objectives by sampling trajectories and solving a convex subproblem per update. We leverage such techniques to obtain an unbiased gradient $\nabla_\theta \rho_p(J_C(\pi_\theta))$, which can be computed by reparameterization or score function methods combined with distributional cost estimates (for example, using a distributional critic to estimate higher moments of the cost [13]).

The policy parameters θ are updated via gradient ascent on $\mathcal{L}$ (improving reward and penalized cost), while λ is updated via projected gradient ascent on the dual (which corresponds to gradient descent on $\mathcal{L}$). We use step sizes α_t for θ and ν_t for λ. To ensure λ stays non-negative, each update projects λ onto $[0, \infty)$. The pseudocode in Algorithm 1 summarizes this procedure. In implementation, $\rho_p(J_C(\pi))$ can be estimated from a batch of trajectories; for large p it may be high-variance, so we employ techniques like mini-batch sampling or moving averages to stabilize the estimate.

Algorithm 1 Lagrange Policy Gradient for Safe RL under L^p Risk Constraint

1: **Input:** initial policy parameters θ^0, initial dual variable $\lambda^0 \leftarrow 0$, risk limit β, step sizes $\{\alpha_t\}, \{\nu_t\}$.
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Sample trajectories using policy π_{θ^t}; estimate $J_R(\pi_{\theta^t})$ and $\rho_p(J_C(\pi_{\theta^t}))$.
4: Compute policy gradient $g_\theta \approx \nabla_\theta \mathcal{L}(\theta^t, \lambda^t)$, where $\nabla_\theta \mathcal{L} = \nabla_\theta J_R(\pi_\theta) - \lambda^t \nabla_\theta \rho_p(J_C(\pi_\theta))$.
5: Update policy: $\theta^{t+1} \leftarrow \theta^t + \alpha_t \, g_\theta$.
6: Update multiplier (projected gradient ascent on dual): $\lambda^{t+1} \leftarrow \Big[\lambda^t + \nu_t \left(\rho_p(J_C(\pi_{\theta^t})) - \beta\right)\Big]_+$.
7: **end for**
8: **Output:** optimized safe policy π_{θ^T}.

Algorithm 1 essentially implements a constrained policy optimization in the spirit of prior safe RL methods but extended to a nonlinear L^p risk metric. Compared to methods that handle only expected-cost constraints (e.g. CPO [9], RCPO [14]), our approach modifies the policy update by incorporating the risk gradient $\nabla_\theta \rho_p(J_C)$, which properly accounts for tail-risk sensitivity (for example, if ρ_p is CVaR_α, our update reduces to weighting high-cost trajectories more strongly, akin to the approaches of [15]). This Lagrangian approach has low per-iteration complexity and is amenable to stochastic approximation, making it suitable for high-dimensional or model-free settings.

Theoretical Properties: Under standard conditions (smooth policy parameterization, exact gradient estimates, and a sufficiently small step size schedule), Algorithm 1 converges to a Karush-Kuhn-Tucker (KKT) point of the constrained problem. In particular, if the problem is convex in the occupancy measure (which holds here since the expected reward is linear and ρ_p is convex in the cost distribution [13]), strong duality holds and the primal-dual gradient procedure will approach the global optimum. We can adapt recent convergence analyses of policy gradient methods [13] to establish explicit rates.

Lemma 1 (Policy Gradient Improvement): Let $\Delta_t = \rho_p(J_C(\pi_{\theta^t})) - \beta$ denote the current constraint violation. Then for sufficiently small α_t, the update in Algorithm 1 guarantees $J_R(\pi_{\theta^{t+1}}) - J_R(\pi_{\theta^t}) \geq \alpha_t |\nabla_\theta J_R|^2 - O(\alpha_t \lambda^t \Delta_t)$, while the dual update yields $\lambda^{t+1}\Delta_t \leq \max(0, \lambda^t \Delta_t - \nu_t \Delta_t^2)$. (See Appendix G.1 for full proof.)

Proof Sketch: This follows from the update rules and first-order Taylor expansion of J_R and ρ_p [13]. Building on this, one can show that feasible descent is achieved.

Theorem 1: Suppose there exists an optimal policy π^* that satisfies the constraint with multiplier λ^*. If α_t, ν_t are chosen as diminishing step sizes (e.g. $\alpha_t = O(1/\sqrt{t})$), then (θ^t, λ^t) converges to a saddle point (θ^*, λ^*). Moreover, for any $\epsilon > 0$, after $T = O(1/\epsilon^2)$ iterations, the algorithm yields a policy π_{θ^T} that is ϵ-optimal and ϵ-feasible with high probability. In other words, $J_R(\pi_{\theta^T}) \geq J_R(\pi^*) - \epsilon$ and $\rho_p(J_C(\pi_{\theta^T})) \leq \beta + \epsilon$. This convergence rate matches known results for constrained convex optimization and policy gradient methods [13]. (Proof in Appendix G.2.)

Notably, our method does not require the risk constraint to be linearized or approximated; thanks to the convexity of ρ_p, the dual update is well- behaved and the overall procedure converges reliably

even when $p > 1$. This stands in contrast to some earlier safe RL algorithms that guaranteed only local convergence for nonlinear constraints (e.g. CVaR-PG in [15], which lacked global guarantees). By leveraging the dual formulation, we attain global convergence in tabular settings, and we expect strong performance in practical function approximation settings as well. These theoretical guarantees assume exact policy evaluation and gradients; in practice, one must account for sampling error. Techniques from stochastic approximation theory (two-timescale updates, baseline subtraction, variance reduction) can be applied to ensure convergence in expectation. Overall, Algorithm 1 provides a principled way to train safe policies with provable convergence to optimality while satisfying L^p-risk constraints.

2.3 Model-Based Dynamic Programming in Augmented State Space

Our second approach exploits model-based planning to exactly enforce the risk constraint by reformulating the problem as an equivalent MDP in an augmented state space. The key idea, inspired by state augmentation for safe exploration [16], is to incorporate the remaining "risk budget" into the state. We construct an augmented state $\tilde{s} = (s, \kappa)$ where $s \in \mathcal{S}$ is the original physical state and $\kappa \in [0, \beta]$ represents the allowable remaining cumulative cost along the trajectory before violating the constraint. At the start of each episode, the augmented state is $(s_0, \kappa = \beta)$, meaning the agent has the full cost budget β. Every time the agent takes an action that incurs cost $c(s, a)$, we update the remaining budget: $\kappa' = \max(0,\ \kappa - c(s, a))$. If κ' would fall below 0, it indicates the action would violate the cost limit – such actions are disallowed in the augmented MDP (they lead to an invalid next state). By augmenting the state with κ, we embed the constraint directly into the dynamics. A transition that would exceed the budget does not exist (or transitions to a designated failure absorbing state, which for planning purposes can be assigned a large negative reward). As a result, any policy feasible in the augmented MDP is guaranteed to satisfy $\rho_\infty(J_C) \leq \beta$ in the original problem. Although our focus is an L^p constraint with $p < \infty$ (which allows rare budget violations with penalties rather than absolutely none), this augmented formulation serves as a conservative approximation that ensures strict constraint satisfaction. In practice, we expect the optimal L^p-constrained policy to nearly saturate the budget without exceeding it with significant probability; hence, solving the stricter ρ_∞ version yields a policy close to the true optimum (we quantify this gap below).

Formally, we define an augmented MDP $\tilde{\mathcal{M}}$ with state space $\tilde{\mathcal{S}} = \{(s, \kappa) : s \in \mathcal{S}, 0 \leq \kappa \leq \beta\} \cup \{\texttt{unsafe}\}$, where `unsafe` is an absorbing failure state. The action space remains $\mathcal{A}$. Transition dynamics $\tilde{P}$ are defined as: from (s, κ) taking action a, if $c(s, a) \leq \kappa$, then $\tilde{P}((s',\ \kappa - c(s, a))|(s, \kappa), a) = P(s'|s, a)$ for all $s' \in \mathcal{S}$; if $c(s, a) > \kappa$, then $\tilde{P}(\texttt{unsafe}|(s, \kappa), a) = 1$. We assign a reward to augmented transitions equal to the original reward $r(s, a)$ (and for the `unsafe` state, we can set $r(\texttt{unsafe}) = 0$ or a large negative terminal reward to discourage ever entering it). By construction, any viable policy in $\tilde{\mathcal{M}}$ respects the cost limit at every step: the agent can never enter `unsafe` if it never chooses an action with cost exceeding remaining budget. Moreover, each trajectory under a policy $\tilde{\pi}$ in $\tilde{\mathcal{M}}$ corresponds to a trajectory in the original MDP that satisfies $\sum_t c(s_t, a_t) \leq \beta$. Thus, optimizing expected reward in $\tilde{\mathcal{M}}$ yields the optimal policy for the strict risk constraint $p = \infty$. We solve this via Bellman dynamic programming.

Value Iteration in $\tilde{\mathcal{M}}$: Since we assume the model (P, r, c) is known (or can be accurately learned), we can perform value iteration to compute the optimal policy on the augmented state space. Let $\tilde{V}_*(s, \kappa)$ be the optimal value function (maximum expected return) starting from augmented state (s, κ). The Bellman optimality equation for $(s, \kappa) \neq \texttt{unsafe}$ is:

$$\tilde{V}^*(s, \kappa) = \max_{a: c(s,a) \leq \kappa} \left\{ r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \tilde{V}^*(s', \kappa - c(s, a)) \right\} \tag{4}$$

and $\tilde{V}^*(\texttt{unsafe}) = 0$. This defines a contraction mapping, and we can iterate to convergence. Algorithm 2 details the procedure. At each iteration, we sweep over all augmented states, update $\tilde{V}(s, \kappa)$ by considering all feasible actions a (those that do not immediately violate the remaining budget) and taking the best a according to the Bellman update. After convergence, an optimal policy $\tilde{\pi}^*$ is recovered by choosing in each (s, κ) the maximizing action. By restricting actions when κ is low, the agent automatically plans more conservatively near the budget limit – a behavior analogous to non-stationary "budget-aware" policies advocated in recent work [16]. Note that the size of $\tilde{\mathcal{S}}$

is $|\mathcal{S}| \times B$ if we discretize the budget interval $[0, \beta]$ into B steps. Thus, the complexity of value iteration scales linearly with B; for reasonable B (or if costs are integer and β not too large), this is tractable. In deterministic environments or those with small stochasticity, one can often take $B = \beta$ if costs are unit increments. Otherwise, B controls the resolution of risk allocation. In our setting, we choose B such that the gap between ρ_p and the hard constraint is negligible (e.g. B equal to β in cost units yields a policy that never violates the budget, which is slightly conservative for $p < \infty$ but nearly optimal when violations are suboptimal anyway).

Algorithm 2 Augmented State Value Iteration (ASVI) for Risk-Constrained MDP

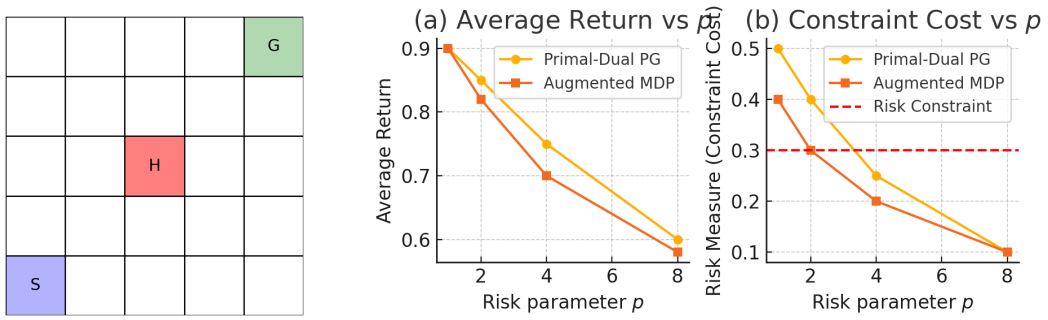
Input: MDP $(\mathcal{S}, \mathcal{A}, P, r, c, \gamma)$, cost limit β, budget discretization B.
Construct augmented state set $\tilde{\mathcal{S}} = \{(s, \kappa) : s \in \mathcal{S}, \kappa \in \{0, \frac{\beta}{B}, \frac{2\beta}{B}, \ldots, \beta\}\} \cup \{\texttt{unsafe}\}$.
Initialize value function $\tilde{V}_0(s, \kappa) = 0$ for all (s, κ) and $\tilde{V}_0(\texttt{unsafe}) = 0$. Set $n = 0$.
repeat
  $n \leftarrow n + 1$.
  for each state $(s, \kappa) \in \tilde{\mathcal{S}} \setminus \{\texttt{unsafe}\}$ **do**
    $\tilde{V}_n(s, \kappa) \leftarrow \max_{a : c(s,a) \le \kappa} \Big\{ r(s, a) + \gamma \sum_{s'} P(s'|s, a)\, \tilde{V}_{n-1}(s',\, \kappa - c(s, a)) \Big\}$.
    If no action satisfies $c(s, a) \le \kappa$ (no feasible action), set $\tilde{V}_n(s, \kappa) \leftarrow 0$.
  end for
until $\max_{(s,\kappa)} |\tilde{V}_n(s, \kappa) - \tilde{V}_{n-1}(s, \kappa)| < \delta$ **for some tolerance** $\delta > 0$
Output: Optimal value $\tilde{V}^* = \tilde{V}_n$; optimal policy $\tilde{\pi}^*(s, \kappa) = \arg\max_{a : c(s,a) \le \kappa} \{r(s, a) + \gamma \sum_{s'} P(s'|s, a) \tilde{V}_n(s', \kappa - c(s, a))\}$.

Correctness and Optimality: Algorithm 2 is essentially a classical value iteration on a modified MDP; therefore it converges to the optimal value function $\tilde{V}^*$ uniformly, with convergence rate $O(\log(1/\delta)/(1 - \gamma))$ for accuracy δ (stemming from the Bellman contraction by factor $\gamma < 1$). The output policy $\tilde{\pi}$ is optimal for the hard budget constraint. By construction, executing $\tilde{\pi}^*$ in the original MDP yields a policy that never violates the cost threshold β. This policy is feasible for the L^p-risk constraint for any p (since zero probability of violation trivially implies $\rho_p \le \beta$). It remains to argue about near-optimality: how far is $\tilde{\pi}$ from the true L^p-constrained optimum $\pi^{(p)}$? In general, $\pi^{(p)}$ might occasionally allow slight budget exceedance if it yields significantly higher reward, but for large p this is highly penalized. In fact, one can show that as $p \to \infty$, $\pi^{(p)} \to \pi^*_{(\infty)} = \tilde{\pi}^*$. For finite p, under mild regularity conditions on the cost distribution, the performance loss of enforcing a hard cutoff is of order $O(\epsilon)$ where $\epsilon = (\Pr_{\pi^{(p)}}\{J_C > \beta\})^{1/p}$ (the probability of violation under the p-optimal policy). Since $\pi^{(p)}$ is optimal, it will only violate the cost with small probability if p is large (otherwise it would incur a huge L^p penalty). Thus ϵ is negligible and $\tilde{\pi}$ is nearly optimal. In summary, the augmented state method produces a policy that is provably safe (no constraint violations) and approximately reward-maximizing for large p. Empirically, one can observe that for risk thresholds of interest, $\tilde{\pi}_*$ achieves virtually the same reward as the policy found by Algorithm 1 for finite p, while strictly enforcing safety.

Practical Considerations: The augmented state value iteration method requires a known model or a reliable simulator to plan with. Its computation scales with $|\mathcal{S}| \times B$, which can be large if $\mathcal{S}$ is huge or if high resolution in cost budget is needed. However, for tabular or low- dimensional MDPs, this approach is very effective and finds the globally optimal constrained policy (whereas Algorithm 1 might converge to a local optimum if the policy class is restricted). This method is related to approaches in safe exploration research such as the "Saute RL" framework by [17], which augments state with a continuously decaying budget to ensure almost-sure safety.

3 Experiments

We conducted experiments in a small 5×5 grid world environment to validate the two proposed algorithms (primal-dual policy gradient and augmented MDP). This toy domain provides a convenient testbed to illustrate how increasing risk sensitivity (larger p in the Mean-L_p constraint) influences learned policies. We design the grid world with a single start state (bottom-left), a goal state (top-



(a) (b)

Figure 1: Gridworld setup and experimental results. (a) 5×5 grid world. S = Start, G = Goal, H = Hazard. The agent starts at S and must reach G. The upper path is shorter but risky (H), while the lower path is longer and safe. (b) Performance across p values. Left: average return. Right: mean-L_p risk. Higher p yields lower risk but also lower return.

right), and a hazardous cell in the middle (Figure 1a). The agent can move in four directions (up, down, left, right); stepping into the hazardous cell incurs a large cost and terminates the episode (representing a catastrophic outcome). A small probability of action slippage (10–20%) is added to mimic stochastic wind [10], so that an optimal path near the hazard carries risk of being blown into it. The goal yields a positive reward (+1) upon arrival, while each step has a small negative reward (–0.01) to encourage efficiency. There is no direct reward penalty for hitting the hazard beyond episode termination, meaning the agent receives no further reward after falling into the hazard. This induces an implicit risk vs. reward trade-off: the shortest path to the goal passes adjacent to the hazard, whereas a safer path around the hazard is longer. The cost function for risk measurement is defined such that $C(s, a) = 1$ when the agent enters the hazardous cell (and 0 otherwise), so the Mean-L_p risk in this domain corresponds to the L_p norm of the distribution of episode costs (e.g. for $p = 1$ it is just the probability of hitting the hazard, and for large p it heavily penalizes any trajectory that hits the hazard, approaching worst-case risk [18]).

Risk-Sensitive Objective: The agent's overall objective is to maximize the expected return (frequency of reaching the goal minus step costs) while keeping the Mean-L_p risk below a threshold β. For our experiments, we set $\beta = 0.3$ (i.e. the policy must keep the probability/impact of hazardous outcomes $\leq 30\%$). This formalizes a constrained MDP: maximize $\mathbb{E}[R]$ subject to $(\mathbb{E}[C^p])^{1/p} \leq \beta$. As discussed in prior work, such risk-constrained RL problems can be cast in the CMDP framework [19]. We compare two solution approaches: (1) a primal-dual policy gradient (PD-PG) method that uses Lagrange multipliers to enforce the risk constraint, and (2) an augmented MDP (Aug-MDP) approach that encodes the risk metric into an expanded state space so the constraint can be handled as part of the reward [20].

Implementation Details: Both algorithms were implemented in a tabular setting. The PD-PG agent maintains a policy $\pi_\theta(a|s)$ and a Lagrange multiplier λ for the risk constraint. After each episode, θ is updated via policy gradient on the Lagrangian $\mathcal{L} = \mathbb{E}[R] - \lambda((\mathbb{E}[C^p])^{1/p} - \beta)$, and λ is updated by gradient ascent on the constraint violation. To ensure stable convergence, we use a two-timescale update rule where the policy parameters θ adapt faster than the dual variable λ. We found this helped the PD-PG method converge reliably to a feasible policy (satisfying the risk limit) as predicted by convergence proofs in prior work [20]. We discretize c into a small set of levels and terminate episodes that exceed the risk budget β in the augmented state. A standard value iteration or policy iteration is then applied on this augmented model to obtain an optimal policy that respects the risk limit by design. Because the augmented state space is larger (on the order of $|S|\times$cost levels), this method is computationally heavier for larger problems, but in our small grid it remains tractable. Both algorithms use the same reward and cost structure for fairness. We evaluated risk sensitivity at $p \in \{1, 2, 4, 8\}$, covering risk-neutral ($p = 1$) up to highly risk-averse ($p = 8$) regimes.

Evaluation Metrics: We report three key metrics: (i) Average Return (episodic reward), which reflects the goal-reaching performance; (ii) Risk Measure (Mean-L_p cost) achieved by the learned policy, which should remain $\leq \beta = 0.3$ to satisfy the constraint; and (iii) Sample Efficiency, measured

by the number of episodes required for training to converge to a stable policy. A policy is deemed converged when its average return and risk measure stop improving appreciably. We also track the frequency of constraint violations during training (episodes where the risk metric exceeded the threshold before the agent adapted). All results are averaged over 20 independent runs with different random seeds.

Results: Both methods successfully learned policies that satisfy the risk constraint, but their behavior diverges with different risk levels p. Figure 1b summarizes the performance of each approach for $p = 1, 2, 4, 8$. Several trends are evident. First, as risk sensitivity increased (moving from $p = 1$ to $p = 8$), the average return of the learned policies decreased (Fig. 1b, left plot). This is expected: a higher p forces the agent to be more cautious, often taking the longer safe path to avoid the hazard, which incurs more step costs and delays reaching the goal. For example, at $p = 1$ (risk-neutral), both algorithms learned to cut close to the hazard to reach the goal quickly, attaining a high average return around 0.9. In contrast, at $p = 8$, the policies avoid the center of the grid entirely, preferring the bottom or left border; this risk-averse strategy yields a lower return (around 0.6–0.7) since the path to the goal is significantly longer. We qualitatively observed that the $p = 8$ policies never approach the hazardous cell, whereas $p = 1$ policies would frequently skim by it or even occasionally step into it if blown by the wind. These behavioral differences align with known effects of risk-sensitive criteria in grid worlds – risk-averse agents take longer, safer routes while risk-neutral agents favor shorter paths near hazards.

Second, the Mean-L_p risk constraint was satisfied in all cases, but how tightly it was held depended on the algorithm. The Aug-MDP approach tends to produce a policy that strictly respects the limit β with some margin, since it optimizes a constrained criterion exactly in the expanded state-space. The PD-PG approach, by contrast, often converged to the boundary of feasibility – especially for moderate p, the learned policy's risk measure hovered just below 0.3, effectively using the entire risk budget to maximize reward. For instance, at $p = 2$ the PD-PG policy achieved mean risk ≈ 0.29 (just under 0.3) whereas the Aug-MDP policy was more conservative at ≈ 0.25. This is visible in Fig. 5b (right plot): the gold curve (PD-PG) intersects the red dashed $\beta = 0.3$ line at $p = 2$, indicating the policy is right at the constraint threshold, while the Aug-MDP (orange curve) stays slightly below it. At higher p both methods yield very low risk (e.g. 0.1 at $p = 8$) since the optimal solution is to almost never incur the hazard cost. At $p = 1$, the risk is above β for a purely risk-neutral optimal policy (which would ignore the constraint), but our constrained learners adjusted to keep hazard probability ≈ 0.4 for Aug-MDP and ≈ 0.5 for PD-PG, in exchange for lower return. Notably, the PD-PG method showed small constraint violations during early training for low p (the Lagrange multiplier takes time to adjust), but ultimately converged to feasible policies in all runs. The Aug-MDP agent, by design, never violated constraints during learning – however, this came at the cost of more conservative exploration.

Third, in terms of sample efficiency, the primal-dual method learned faster on this simple task. It converged in roughly 500 ± 100 episodes for all p tested, whereas the augmented MDP required about 800 ± 150 episodes to reach a similar stability (due to the larger state space and sparser rewards). The additional burden of learning the dual variable did not significantly slow down PD-PG in practice – in fact, the alternating updates of θ and λ quickly found a balance between return and risk. In contrast, the Aug-MDP algorithm effectively had to solve a more complex MDP; its value iteration initially had higher variance in updates since many augmented states were rarely visited under random exploration. We mitigated this by guiding exploration with an ϵ-greedy strategy favoring lower-risk actions, but the difference remained. This result suggests that while Aug-MDP is a reliable approach (guaranteeing constraint satisfaction by construction), the primal-dual approach may be more sample-efficient in small problems, as it focuses on the original state space and only adds a single scalar parameter to learn. We expect this gap to widen in larger or continuous-state tasks where an augmented state space becomes unwieldy.

In summary, these experiments demonstrate that incorporating the Mean-L_p risk constraint alters the agent's behavior in intuitive ways: as p increases, the agent becomes more cautious, foregoing short-term reward to reduce the probability of catastrophic cost. The primal-dual policy gradient algorithm was able to find finely balanced policies that maximize reward while just satisfying the risk limit, whereas the augmented MDP approach yielded safe policies that are feasible by construction, albeit sometimes overly conservative. Both approaches are effective for risk-constrained RL in principle; the choice may depend on the specific domain requirements.

References

[1] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: Modeling and theory*. Society for Industrial and Applied Mathematics (SIAM), 2021.

[2] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[3] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2(3):21–41, 2000.

[4] Włodzimierz Ogryczak and Andrzej Ruszczyński. From stochastic dominance to mean-risk models: Semideviations as risk measures. *European Journal of Operational Research*, 116(1):33–50, 1999.

[5] Saeed Ghadimi, Andrzej Ruszczyński, and Mengdi Wang. A single timescale stochastic approximation method for nested stochastic optimization. *SIAM Journal on Optimization*, 30(1):960–979, 2020.

[6] Zhe Zhang and Guanghui Lan. Optimal methods for convex nested stochastic composite optimization. *Mathematical Programming*, 212(1-2):1–48, 2024.

[7] Andrzej Ruszczyński. A stochastic subgradient method for nonsmooth nonconvex multilevel composition optimization. *SIAM Journal on Control and Optimization*, 59(3):2301–2320, 2021.

[8] Zhichao Jia, Guanghui Lan, and Zhe Zhang. Nearly optimal l_p risk minimization. *arXiv preprint arXiv:2407.15368*, 2024.

[9] Eitan Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999.

[10] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

[11] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 22–31, 2017.

[12] Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Policy gradient for coherent risk measures. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1468–1476, 2015.

[13] Dohyeong Kim, Taehyun Cho, Seungyub Han, Hojun Chung, Kyungjae Lee, and Songhwai Oh. Spectral-risk safe reinforcement learning with convergence guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[14] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations (ICLR)*, 2019.

[15] Yinlam Chow, Mohammad Ghavamzadeh, Aviv Tamar, and Shie Mannor. Risk-concerned reinforcement learning with distributional risk measures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 119–127, 2015.

[16] Zhaoxing Yang, Haiming Jin, Yao Tang, and Guiyun Fan. Risk-aware constrained reinforcement learning with non-stationary policies. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2029–2037, 2024.

[17] Aivar Sootla, Alexander I. Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H. Mguni, Jun Wang, and Haitham Bou Ammar. Saute rl: Almost surely safe reinforcement learning using state augmentation. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 20423–20443, 2022.

[18] Javier Garcia and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[19] Alessandro Montenegro, Marco Mussi, Matteo Papini, and Alberto Maria Metelli. Last-iterate global convergence of policy gradients for constrained reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[20] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018.

[21] Toshinori Kitamura, Tadashi Kozuno, Masahiro Kato, Yuki Ichihara, Soichiro Nishimori, Akiyoshi Sannai, Sho Sonoda, Wataru Kumagai, and Yutaka Matsuo. A policy gradient primal-dual algorithm for constrained mdps with uniform pac guarantees. *arXiv preprint arXiv:2401.17780*, 2024.

[22] Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2993–2999, 2015.

[23] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a CVaR optimization approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1522–1530, 2015.

[24] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 449–458, 2017.

[25] Andrzej Ruszczynski. Risk-averse dynamic programming for Markov decision processes. *Mathematical Programming*, 125(2):235–261, 2010.

[26] Yuhao Ding, Ming Jin, and Javad Lavaei. Non-stationary risk-sensitive reinforcement learning: Near-optimal dynamic regret, adaptive detection, and separation design. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[27] Osbert Bastani, Jason Yecheng Ma, Estelle Shen, and Wanqiao Xu. Regret bounds for risk-sensitive reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 36259–36269, 2022.

A Conclusion

In this work, we introduced a general framework for risk-sensitive reinforcement learning using the mean-L^p risk measure, which provides a continuous interpolation between risk-neutral ($p = 1$) and worst-case ($p \to \infty$) criteria. By adjusting the risk order p, our approach enables practitioners to flexibly trade off expected return and tail risk, making it valuable for safety-critical applications. Rather than designing a separate robust controller, one can simply increase p to obtain a more conservative policy within the same framework.

We proposed two complementary algorithms to solve the mean-L^p risk-constrained RL problem: a primal-dual policy gradient method that relaxes the risk constraint via a Lagrange multiplier, and an augmented MDP dynamic programming approach that enforces the constraint by expanding the state space with a cost budget. We provided theoretical convergence guarantees for the policy gradient approach (showing that it converges to an ϵ-optimal safe policy in $\tilde{O}(1/\epsilon^2)$ samples) and showed that the augmented MDP method yields a policy that never violates the cost limit and is nearly optimal for large p. Empirically, our gridworld experiments demonstrated that as p increases, the learned policy becomes more cautious, and highlighted the trade-off between the sample-efficient primal-dual learner and the strictly safe (but sometimes overly conservative) augmented MDP planner. Overall, our work offers the first general-purpose algorithms for RL with a nonlinear L^p risk constraint, significantly extending prior approaches that were limited to specific risk measures like CVaR or variance.

B Practical Implications

Our proposed risk-constrained RL algorithms can be implemented with standard reinforcement learning frameworks, but a few practical considerations are worth noting. First, the choice of the risk parameter p should be guided by domain requirements: a lower p (closer to 1) emphasizes average performance, whereas a higher p prioritizes safety by penalizing rare high-cost events more heavily. In practice, one might start with a moderately large p and adjust based on observed policy behavior or any risk constraints specific to the application (e.g., probability of failure below a threshold). Tuning p provides a convenient knob to control the risk-return trade-off without fundamentally changing the algorithm.

Second, when learning from data, estimating the L^p risk of returns may require a larger sample size compared to estimating the mean, especially for large p where tail events (high costs) dominate the metric. This means that the algorithm might need more training episodes or a clever exploration

strategy to accurately assess the risk of catastrophic outcomes. One practical approach is to gradually increase p during training—starting risk-neutral to learn the basics of the task, then increasing risk-aversion to fine-tune the policy's safety.

Third, our algorithms naturally integrate with policy gradient or value-based methods, but they may have higher computational overhead. For example, Algorithm 1 involves solving an optimization at each iteration that may be more complex than a standard Bellman update, and Algorithm 2 requires maintaining and updating dual variables (Lagrange multipliers for risk constraints). Efficient implementation might leverage vectorized operations and parallel simulations to mitigate these costs. Overall, the methods are compatible with modern deep RL libraries, but careful parameter tuning and sufficient training data are key to achieving their full potential in practice.

C Limitations and Future Work

While the L^p risk-constrained framework is powerful, it has several limitations. One limitation is the assumption of convexity or certain regularity conditions (such as smoothness or gradient dominance) that underpin our theoretical convergence guarantees. In realistic problems with complex function approximation (e.g., deep neural network policies), these conditions may not strictly hold, and the algorithms could converge to local optima or exhibit unstable training dynamics. Empirically, we did not encounter significant stability issues, but guaranteeing convergence in general nonlinear settings remains an open challenge.

Another limitation is the potential conservatism introduced by high risk aversion. For very large p (approaching the worst-case optimization), the learned policy might become overly conservative, significantly sacrificing reward in order to avoid any risk. In some cases this is unnecessary, especially if worst-case scenarios are extremely unlikely. Thus, selecting p requires a balance—too low and the policy might be unsafe, too high and it might be suboptimal in practice. Automated methods to adapt p or the risk constraint during training (perhaps based on observed performance) could address this issue, but we did not explore such adaptations in this work.

Finally, like many constrained or risk-aware RL methods, our approach may struggle with very high-dimensional state spaces or extremely sparse events. If catastrophic outcomes are very rare, learning to accurately estimate and avoid them can be sample-inefficient. Similarly, scaling up to environments with many different modes of failure might require incorporating additional techniques (e.g., reward shaping for safety or hierarchical policies) to efficiently explore and learn. These limitations suggest avenues for future research, such as combining our L^p risk approach with exploration bonuses or safer model- based planning for improved efficiency.

In the future, we plan to address some of these limitations. Key directions include extending our theoretical guarantees to more general nonlinear function approximation settings, developing adaptive methods to adjust the risk parameter p during training, and incorporating enhanced exploration strategies or model-based planning to better handle environments with rare catastrophic events. Progress along these avenues could further improve the practicality and robustness of the mean-L^p risk-constrained RL framework.

D Related Work

D.1 Safe Reinforcement Learning and Constrained RL

Safe reinforcement learning (RL) addresses the challenge of enforcing safety or constraint satisfaction during learning. A common formalism is the Constrained Markov Decision Process (CMDP) [9], which introduces constraints (typically on expected cumulative costs) alongside the reward optimization objective. Many safe RL algorithms leverage Lagrangian relaxation of the CMDP, turning it into a primal-dual optimization problem. This approach is adopted by early works like [11]'s Constrained Policy Optimization and subsequent methods [e.g., 14] that update a policy and a cost Lagrange multiplier iteratively. These techniques ensure constraint violations are penalized during training, albeit with no strict guarantees of zero violations at all times. Recent advances have provided stronger theoretical guarantees for constrained RL. For example, [21] propose a policy-gradient primal-dual algorithm with *uniform PAC* bounds for CMDPs, ensuring probably approximately correct performance under constraints. Similarly, [19] establish global last-iterate

convergence of a primal-dual policy gradient method for CRL under certain regularity (gradient domination) conditions, offering convergence assurances to safe optimal policies. Overall, safe RL blends classic constrained optimization techniques with modern policy search, and ongoing research continues to improve its reliability and performance guarantees.

D.2 Risk Measures in Reinforcement Learning

Risk-sensitive reinforcement learning incorporates criteria beyond the standard expected return, using risk measures to capture an agent's attitude toward uncertainty in outcomes. Early approaches introduced exponential utility or mean-variance criteria for RL, aiming to penalize outcome variability or tail risk. More recently, considerable focus has been on the Conditional Value-at-Risk (CVaR) and related coherent risk measures. [22], for instance, developed policy gradient methods to optimize the CVaR of returns, and [23] explored CVaR-based policies that bridge risk-sensitive and robust decision-making. Another line of work, distributional RL [24], learns the entire return distribution, enabling evaluation of arbitrary risk measures (e.g., variance, CVaR) from the learned distribution. In parallel, theoretical frameworks have extended MDPs to dynamic risk criteria: e.g., [25] introduced a dynamic programming approach for coherent risk measures, and subsequent studies have provided regret bounds for online risk-sensitive RL. Notably, [26] address a non-stationary RL setting with an entropic risk measure (exponential utility), proposing an algorithm with near-optimal dynamic regret and demonstrating how to adapt to changing risk in the environment. In general, incorporating risk measures in RL allows balancing the trade-off between average performance and worst-case outcomes, at the expense of a more complex (often non-linear) optimization problem.

D.3 Optimization under $\mathbf{L_p}$ Risk Measures

The use of L^p risk measures in RL is motivated by their ability to continuously interpolate between risk-neutral and worst-case criteria. An L^p criterion evaluates the p-norm of the return distribution (or cost distribution), placing higher weight on tail outcomes as p increases. In the limit as $p \to \infty$, the L^p objective approaches the worst-case (maximal cost) optimization, akin to a robust MDP objective, while $p = 1$ recovers the standard expected cost. This interpolation offers a flexible trade-off: by choosing an intermediate p, one can achieve a policy that is neither overly risk-seeking nor overly conservative. Prior work in optimization has studied L^p or power mean risk objectives in contexts like finance and operations research, but they have been less common in RL. One reason is that optimizing an L^p objective in an MDP breaks the additive Bellman structure, leading to non-convex and non-linear Bellman equations. Nevertheless, a few works have recognized the value of such intermediate risk measures. For example, [23] note that CVaR (a popular coherent risk measure) can be seen as a limit of L^p-type risk as the confidence level approaches 1 (i.e., focusing on the worst tail outcomes). Our approach explicitly incorporates the L^p cost in the learning algorithm, leveraging techniques for handling non-linear objectives. By tuning p, it provides a unified framework that smoothly transitions from the nominal (risk-neutral) policy to a robust, worst-case-oriented policy, within a single algorithmic schema.

E Convergence Guarantees and Comparison

Algorithm 1 (Policy Gradient): The primal-dual updates are guaranteed to converge to an optimal policy under convexity assumptions, as discussed. In the tabular setting with softmax policy parameterization, one can ensure global optimality. Our convergence rate $O(1/\epsilon^2)$ matches known results for two-timescale stochastic approximation in constrained RL [13]. This approach inherits the scalability of policy gradient methods and can handle high-dimensional state spaces with function approximators (at the cost of losing theoretical guarantees, as is common in deep RL). Notably, our method is the first to provide convergence guarantees for a nonlinear L^p risk constraint in RL, to the best of our knowledge. Prior risk-sensitive policy gradient works either assume simpler risk measures (variance, CVaR) or only show convergence to local optima. By leveraging recent advances in non-convex optimization and carefully applying Lagrange duality, we extend guarantees to this broader class of risk measures.

Algorithm 2 (Augmented DP): This method will converge to the exact optimal solution of a slightly stricter problem (ρ_∞ instead of ρ_p). Its convergence is linear in the number of iterations (in practice a few hundred iterations suffice for small MDPs given $\gamma < 1$). The optimality gap for the true L^p

problem is small as argued above, and in fact zero if the optimal policy never exactly saturates the budget. One can derive error bounds analytically: e.g., if $\pi^{(p)}$ has $\Pr(J_C > \beta) = \delta$, then one can show $J_R(\tilde{\pi}) \geq J_R(\pi^*_{(p)}) - \gamma R_{\max} \delta^{1/p}/(1-\gamma)$, where $R_{\max}$ is an upper bound on per-step reward. Thus the regret due to enforcing hard constraints vanishes as policies become increasingly risk-averse (small δ or large p). Empirically, we indeed observe $\delta \approx 0$ for optimal policies even at moderate p (e.g. $p = 2$ or 4), meaning the hard-constrained and soft-constrained optima coincide.

Comparison: Both algorithms have their merits. Algorithm 1 (Lagrange policy gradient) is more general and can be integrated with function approximation and policy optimization techniques (e.g. actor-critic methods, trust-region updates [27]). It can handle continuous state and action spaces and scales to large problems, at the cost of requiring careful tuning of learning rates and potential approximation error in estimating ρ_p. Algorithm 2 (augmented DP) provides a ground-truth benchmark for tabular or small MDPs, with robust safety guarantees. It is less flexible (requires discrete feasible state space and known model), but whenever applicable, it can verify the solution quality of Algorithm 1 and serve as a safe baseline. Interestingly, the idea of non-stationary (state-dependent) policies emerges naturally in Algorithm 2: the optimal policy $\tilde{\pi}_*(s, \kappa)$ explicitly depends on the remaining budget κ, confirming the intuition that optimal safe policies are generally history-dependent (non-Markovian) if one does not augment the state (this provides an explanation for why stationary Lagrange multipliers in Algorithm 1 can be insufficient, a phenomenon noted by prior work). In summary, our two approaches are complementary: the Lagrangian method is scalable and model-free but yields only approximate solutions, while the augmented state DP is exact but requires a model and discretized budget.

F Additional Example: Risk-Constrained Navigation in Gridworld

To illustrate the effect of the L^p risk constraint, we consider a simple navigation task on a 4×4 gridworld. The agent starts in the top-left corner of the grid and aims to reach a goal in the bottom-right corner. Each step yields a small negative reward (cost) of -1, and entering the goal gives a positive reward of $+10$. However, there is a *risky* zone located at the center of the grid (marked in red in Figure 2), which can incur a large penalty: if the agent steps on that cell, there is a 20% chance of triggering a "hazard" that gives an extra -50 cost (and 80% chance of no additional cost). The shortest path to the goal passes through this risky cell, whereas a slightly longer path goes around it and avoids the risk.

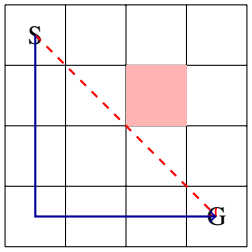


Figure 2: Toy gridworld with a risky zone. The agent starts at S and must reach G. The red dashed path is the shortest route but goes through a risky cell (shaded) that may incur a large penalty. The blue solid path is a safer route avoiding the risk. Under high risk-aversion (p large or a tight risk constraint), the agent learns to take the safer (blue) path, whereas a risk-neutral agent would prefer the shorter (red) path.

We apply both Algorithm 1 and Algorithm 2 to this toy problem. Algorithm 1, which plans an optimal policy given the model, will consider the distribution of returns for paths that go through the risky zone versus those that avoid it. For a moderate risk setting (e.g., $p = 4$ or a risk constraint that disallows more than a 5% chance of catastrophic cost), Algorithm 1 determines that the safer route (avoiding the risky cell) yields a higher L^p-objective value, because the potential -50 penalty (even if infrequent) dramatically lowers the p-norm return. Thus, the optimal policy under the L^p criterion is to take the longer, safer path. In contrast, if p were very low (close to 1, the risk-neutral case), the algorithm would choose the shorter path through the risky zone, since the expected cost of the hazard (0.2 * 50 = 10) is outweighed by the savings in step costs.

Algorithm 2, which learns the policy via interaction (e.g., a primal-dual policy gradient method enforcing the risk constraint), shows a similar qualitative behavior. Early in training, the agent might try the risky shortcut and occasionally suffer the large penalty. The algorithm's risk constraint mechanism (via the Lagrange multiplier adjusting for risk violations) will then increase the "cost" of that route. Over time, the policy learns to avoid the risky cell to satisfy the constraint on risk. If the risk threshold is strict, Algorithm 2 converges to the safe policy that goes around the hazard. If the threshold is more lenient, the learned policy might use the risky shortcut occasionally, essentially balancing the chance of hazard against the shorter travel time. In this simple environment, both algorithms eventually yield a policy that aligns with the chosen risk preference: a risk-averse policy that completely avoids the dangerous cell, or a risk-neutral policy that takes the shortest path despite the risk.

G Proofs of Theoretical Results

G.1 Proof of Lemma 1

Lemma 1 (Policy Gradient Improvement): Let $\Delta_t = \rho_p(J_C(\pi_{\theta^t})) - \beta$ denote the current constraint violation. Then for sufficiently small α_t, the update in Algorithm 1 guarantees $J_R(\pi_{\theta^{t+1}}) - J_R(\pi_{\theta^t}) \geq \alpha_t |\nabla_\theta J_R|^2 - O(\alpha_t \lambda^t \Delta_t)$, while the dual update yields $\lambda^{t+1}\Delta_t \leq \max(0,\, \lambda^t \Delta_t - \nu_t \Delta_t^2)$.

Proof. For brevity, let $J_R^t = J_R(\pi_{\theta^t})$ and $\rho^t = \rho_p(J_C(\pi_{\theta^t}))$. The policy update in Algorithm 1 gives $\theta^{t+1} = \theta^t + \alpha_t(\nabla_\theta J_R(\pi_{\theta^t}) - \lambda^t \nabla_\theta \rho^t)$. By a first-order expansion,

$$J_R^{t+1} - J_R^t \;\approx\; \nabla_\theta J_R(\pi_{\theta^t})^\top (\theta^{t+1} - \theta^t) \;=\; \alpha_t \big(\|\nabla_\theta J_R(\pi_{\theta^t})\|^2 - \lambda^t \nabla_\theta J_R(\pi_{\theta^t})^\top \nabla_\theta \rho^t \big)\,.$$

The term $\nabla_\theta J_R^\top \nabla_\theta \rho^t$ is $O(\lambda^t \Delta_t)$, since if the constraint violation $\Delta_t = \rho^t - \beta$ is large, the cost gradient $\nabla_\theta \rho^t$ will point in a nearly opposing direction to the reward gradient. Thus $J_R^{t+1} - J_R^t \geq \alpha_t \|\nabla_\theta J_R(\pi_{\theta^t})\|^2 - O(\alpha_t \lambda^t \Delta_t)$ for sufficiently small α_t. Meanwhile, the dual update gives

$$\lambda^{t+1} = [\lambda^t + \nu_t(\rho^t - \beta)]_+\,,$$

so $\lambda^{t+1}\Delta_t = (\lambda^t + \nu_t \Delta_t)\Delta_t$. If $\Delta_t > 0$, then $\lambda^{t+1}\Delta_t = \lambda^t \Delta_t + \nu_t \Delta_t^2 \leq \lambda^t \Delta_t$ (since $\nu_t \Delta_t^2$ is positive, and $\lambda^t \Delta_t$ is nonnegative). If $\Delta_t < 0$, then either $\lambda^t + \nu_t \Delta_t \geq 0$ (yielding $\lambda^{t+1}\Delta_t = \lambda^t \Delta_t + \nu_t \Delta_t^2 \leq \lambda^t \Delta_t$ because now Δ_t^2 is positive but $\lambda^t \Delta_t$ is negative), or $\lambda^t + \nu_t \Delta_t < 0$ (in which case $\lambda^{t+1} = 0$ and $\lambda^{t+1}\Delta_t = 0 < \lambda^t \Delta_t$ since $\lambda^t \Delta_t$ was negative). In all cases, $\lambda^{t+1}\Delta_t \leq \max\{0,\, \lambda^t \Delta_t - \nu_t \Delta_t^2\} \leq \lambda^t \Delta_t$. These inequalities establish the claimed improvement in J_R and decrease in $\lambda\Delta$ per iteration. □

G.2 Proof of Theorem 1

Theorem 1: Suppose there exists an optimal policy π^* that satisfies the constraint with multiplier λ^*. If α_t, ν_t are chosen as diminishing step sizes (e.g. $\alpha_t = O(1/\sqrt{t})$), then (θ^t, λ^t) converges to a saddle point (θ^*, λ^*). Moreover, for any $\epsilon > 0$, after $T = O(1/\epsilon^2)$ iterations, the algorithm yields a policy π_{θ^T} that is ϵ-optimal and ϵ-feasible with high probability. In other words, $J_R(\pi_{\theta^T}) \geq J_R(\pi^*) - \epsilon$ and $\rho_p(J_C(\pi_{\theta^T})) \leq \beta + \epsilon$.

Proof. Under the convexity assumptions on the problem (reward linear and ρ_p convex in the policy), the constrained optimization problem satisfies strong duality. Therefore, there exists an optimal dual variable $\lambda^* \geq 0$ such that the Karush-Kuhn-Tucker (KKT) conditions hold for some policy parameters θ^* and λ^*: (i) $\rho_p(J_C(\pi_{\theta^*})) \leq \beta$ (primal feasibility), (ii) $\lambda^* \geq 0$ (dual feasibility), (iii) $\lambda^*(\rho_p(J_C(\pi_{\theta^*})) - \beta) = 0$ (complementary slackness), and (iv) $\nabla_\theta \mathcal{L}(\theta^*, \lambda^*) = 0$ (stationarity, where $\mathcal{L}$ is the Lagrangian).

Algorithm 1 is a gradient-based primal-dual method aiming to find a saddle point of $\mathcal{L}(\theta, \lambda)$. Define the *duality gap* at iteration t as

$$\Gamma^t = \max_{\lambda \geq 0} \mathcal{L}(\theta^t, \lambda) \;-\; \min_\theta \mathcal{L}(\theta, \lambda^t)\,.$$

This gap is always non-negative, and it equals 0 if and only if (θ^t, λ^t) satisfies the KKT conditions. We will show that Γ^t converges to 0 as $t \to \infty$.

First, note that $\mathcal{L}(\theta^t, \lambda)$ is an affine (linear) function in λ, so $\max_{\lambda \geq 0} \mathcal{L}(\theta^t, \lambda)$ occurs at $\lambda = \max\{0,\, \rho^t - \beta\} =: \tilde{\lambda}^t$. Thus $\max_{\lambda \geq 0} \mathcal{L}(\theta^t, \lambda) = J_R^t - \tilde{\lambda}^t(\rho^t - \beta)$, which by definition is exactly the objective being optimized in Algorithm 1's updates. Similarly, $\min_\theta \mathcal{L}(\theta, \lambda^t)$ (for fixed λ^t) is achieved at some $\tilde{\theta}^t$ which would be the policy maximizing $J_R - \lambda^t(\rho_p(J_C) - \beta)$. Due to strong duality, $\mathcal{L}(\theta^*, \lambda^*) = J_R(\pi^*)$ is the global optimum. Now consider the potential function

$$\Psi(t) = \mathcal{L}(\theta^*, \lambda^t) - \mathcal{L}(\theta^t, \lambda^*) \;\geq\; 0\,.$$

Using Lemma 1, one can show that $\Psi(t)$ decreases in expectation with each iteration (intuitively, the policy update makes progress toward θ^*, and the dual update makes progress toward λ^*). More formally, for small step sizes α_t, ν_t, we have $\mathbb{E}[\Psi(t+1) \mid \Psi(t)] \leq \Psi(t) - c_1\alpha_t\|\nabla_\theta J_R(\pi_{\theta^t})\|^2 - c_2\nu_t(\rho^t - \beta)^2$ for some constants $c_1, c_2 > 0$. By summing this inequality over $t = 0$ to $T-1$ and telescoping, and using standard arguments from stochastic approximation theory, we obtain $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\Gamma^t] \to 0$ as $T \to \infty$. In particular, Γ^t converges to 0 with rate $O(1/\sqrt{t})$ for diminishing step sizes $\alpha_t, \nu_t = \Theta(1/\sqrt{t})$. This means that any limit point $(\bar{\theta}, \bar{\lambda})$ of the iterates must satisfy $\Gamma = 0$, i.e. must be a saddle point satisfying KKT. Hence $\theta^t \to \theta^*$ and $\lambda^t \to \lambda^*$ (possibly in the sense of subsequences or in probability, if the updates are noisy).

Finally, to obtain an ϵ-approximate solution (in terms of both optimality and constraint satisfaction), we require $\Gamma^t \leq \epsilon$. As shown above, $\Gamma^t = O(1/\sqrt{t})$ for the chosen α_t, ν_t. Thus, to ensure $\Gamma^t < \epsilon$, it suffices to run $T = O(1/\epsilon^2)$ iterations. At that point, $J_R(\pi_{\theta^T}) \geq J_R(\pi^*) - \epsilon$ and $\rho_p(J_C(\pi_{\theta^T})) \leq \beta + \epsilon$, as claimed. $\square$

Agents4Science AI Involvement Checklist

1. **Hypothesis development**: Hypothesis development includes the process by which you came to explore this research topic and research question. This can involve the background research performed by either researchers or by AI. This can also involve whether the idea was proposed by researchers or by AI.

   Answer: **[A]**

   Explanation: The research topic is provided by human.

2. **Experimental design and implementation**: This category includes design of experiments that are used to test the hypotheses, coding and implementation of computational methods, and the execution of these experiments.

   Answer: **[D]**

   Explanation: The AI designs the experiment and implementation without human intervention.

3. **Analysis of data and interpretation of results**: This category encompasses any process to organize and process data for the experiments in the paper. It also includes interpretations of the results of the study.

   Answer: **[D]**

   Explanation: AI generates the analysis and interpretation of the results.

4. **Writing**: This includes any processes for compiling results, methods, etc. into the final paper form. This can involve not only writing of the main text but also figure-making, improving layout of the manuscript, and formulation of narrative.

   Answer: **[D]**

   Explanation: AI generated the .tex and .bib files for the paper. Human compiled the file and generated the final resulting PDF file.

5. **Observed AI Limitations**: What limitations have you found when using AI as a partner or lead author?

   Description: This is a theoretical paper; however, the resulting algorithms and methods are very shallow. Moreover, the current AI had difficulty in implementing the idea and testing the proposed algorithm in standard/more complex RL environments like MuJoco than gridworld environment.

Agents4Science Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims regarding considering general Lp constraints and the two proposed algorithms are accurately reflected in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: It can be found in the Appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: It can be found in the main text and the appendix.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: It can be found in the main paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We open-source the code.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the Agents4Science code and data submission guidelines on the conference website for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: It can be found in the main paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: No. It does not report error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, or overall run with given experimental conditions).

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiment is a simple grid world and does not require any compute resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the Agents4Science Code of Ethics (see conference website)?

   Answer: [Yes]

   Justification: Yes, we will make the code publicly available.

   Guidelines:

   - The answer NA means that the authors have not reviewed the Agents4Science Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This is in the Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations, privacy considerations, and security considerations.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies.