End-to-End Risk-Aware Planning by Gradient Descent

Noah Patton*, Jihwan Jeong*, Michael Gimelfarb*†, Scott Sanner†
Department of Mechanical & Industrial Engineering, University of Toronto, Canada
noah.patton@mail.utoronto.ca, jhjeong@mie.utoronto.ca, mike.gimelfarb@mail.utoronto.ca, ssanner@mie.utoronto.ca

Abstract

Planning provides a framework for optimizing sequential decisions in potentially complex environments. A recent advance in efficient planning in deterministic high-dimensional domains with continuous action spaces leverages backpropagation through a model of the environment to directly optimize the actions. However, this method does not take risk into account when optimizing decisions in highly stochastic environments. We address this problem by introducing Risk-Aware Planning using PyTorch (RAPTOR), a framework that handles risk in stochastic planning domains through an end-to-end optimization of entropic utility. While we cannot directly formalize the distributionally-defined entropic utility in closed-form for end-to-end planning, in settings where all MDP stochasticity is defined through the location-scale family, we can reparameterize the objective and apply stochastic backpropagation. What is notable in this approach is that the entropic utility is defined based on sufficient statistics computed from forward sampled trajectories, but due to the nature of autodifferentiation, we can still backpropagate through the entropic utility and these sufficient statistics. The resulting sequence of actions, which we call the risk-sensitive straight-line plan, provides a lower bound on the utility of the optimal policy and can be seen as a form of hindsight optimization. We evaluate RAPTOR on two highly stochastic domains, including nonlinear navigation and linear reservoir control, demonstrating the ability to manage risk in complex MDPs.

1 Introduction

As more machine learning models are deployed in the real world, the concern over ensuring their safety has been ever-increasing (Faria 2018; Pereira and Thomas 2020). In sequential stochastic decision-making problems in particular, it has been shown that optimizing the expected cumulative reward can lead to undesirable outcomes such as excessive risk-taking, since low-probability catastrophic outcomes with negative reward, or *risk*, are often underrepresented (Moldovan 2014). The *risk-averse MDP* addresses this problem by optimizing risk measures with favorable mathematical properties (Ruszczyński 2010).

Planning learns optimal decisions or actions given a mathematical description of the environment, thus minimizing the need to do dangerous exploration in the real world. However, despite advances in scalable end-to-end planning techniques, existing approaches do not typically take risk into account. In particular, *BackpropPlan* (Wu, Say, and Sanner 2017) utilizes recent advances in deep learning and is highly scalable in continuous state or action spaces. In this framework, the transition model and reward are encoded in RNN-like cells. Unlike a typical neural network, the inputs to the network are the actions that are optimized through backpropagation. By employing a highly effective non-convex optimizer (Tieleman and Hinton 2012), BackpropPlan is able to efficiently learn optimal sequences of actions in continuous state and action planning domains. However, its main limitation is that it cannot even be applied to stochastic models. This could be addressed by learning *reactive policies* in continuous state and action MDPs (CSA-MDPs) (Bueno et al. 2019); however, this work also does not incorporate risk into the decision-making.

In this paper, we propose RAPTOR (*Risk-Aware Planning using pyTORch*), which enables scalable risk-aware end-to-end planning for continuous state and action MDPs leveraging autodifferentiation (Paszke et al. 2019) for gradient-based optimization w.r.t. the MDP model and an entropic utility objective. To achieve this, we begin by leveraging an extension of BackpropPlan to accommodate stochastic transitions (Bueno et al. 2019), by representing the planning domain as a stochastic computation graph (Section 3.1). While we cannot directly formalize the distributionally-defined entropic utility in closed-form for end-to-end planning, in settings where all MDP stochasticity is defined through the location-scale family, we can reparameterize the objective and apply stochastic backpropagation (Section 3.2). This allows us to compute the entropic utility directly from the sufficient statistics of forward sampled trajectories, while still permitting backpropagation through the entropic utility and its symbolic computation from sufficient statistics of trajectories thanks to autodifferentiation and reparameterization.

The entropic utility objective is a particularly suitable objective function for end-to-end planning, satisfying well-known convexity and recursive properties that we exploit directly in this work to derive a risk-sensitive version of the straight-line plan (Section 3.3). This makes it possi-

*Authors contributed equally.
†Affiliate to Vector Institute, Toronto, Canada.

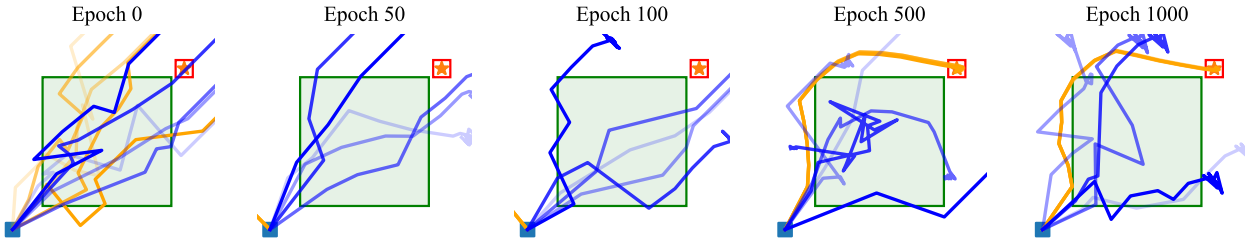

Figure 1: The evolution of trajectories navigated by RAPTOR (black) and a risk-neutral agent (blue) in a two-dimensional Navigation domain subject to stochastic dynamics (note: multiple sample trajectories are shown simultaneously in each epoch). The green bounding box at the center is a high-variance zone. The more an agent traverses into the box the higher the variability in the next position at which the agent lands. The blue square on the bottom left is the starting position, while the red box at the upper right corner shows the goal region. By epoch 1000, we clearly see that the risk-sensitive agent is able to get to the goal region by avoiding the box and thus failure, while the risk-neutral agent does not.

ble to directly optimize risk in sequential decision-making problems without relying explicitly on the Bellman principle, which often presents unique computational challenges in other risk-aware MDP frameworks (Defourny, Ernst, and Wehenkel 2008; Mannor and Tsitsiklis 2011).

While our resulting optimization of the straight-line plan may be sub-optimal in stochastic environments, we prove that the entropic utility of the straight-line plan *lower bounds that of the optimal entropic utility policy*. Indeed, as evidenced by Figure 1, this lower bound is effective in finding risk-averse behaviors in highly stochastic environments in a computationally efficient manner. Overall, empirical evaluations in Section 4 on two highly stochastic domains involving continuous action parameters — navigation (Faulwasser and Findeisen 2009) and reservoir control (Yeh 1985) — demonstrate that RAPTOR provides a reliable and efficient end-to-end method for risk-sensitive planning in complex MDPs.

While the present work focuses solely on risk-aware planning in MDPs, we envision its potential future use as an efficient planning component in a risk-aware model-based reinforcement learning framework. In this case, being risk-aware could offer one way to avoid worst-case outcomes when training a policy on rollouts from an imperfect model of the environment, and deploying this policy to a real-world setting.

2 Preliminaries

2.1 Markov Decision Processes

Sequential decision-making problems in this work are modeled as *continuous state-action Markov decision processes* (CSA-MDPs), defined as tuples $\langle \mathcal{S}, \mathcal{A}, r, p, \mathbf{s}_0 \rangle$: $\mathcal{S} \subseteq \mathbb{R}^n$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^m$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a bounded differentiable reward function, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ describes the non-linear dynamics of the system, and $\mathbf{s}_0$ is the initial state. Note that CSA-MDPs are naturally factored (Boutilier, Dean, and Hanks 1999), such that the state components are mutually independent given the previous state $\mathbf{s}_t$ and action $\mathbf{a}_t$. A *policy* is a mapping $\pi_t : \mathcal{S} \to \mathcal{A}$, while a *plan* is an explicit sequence of actions $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_H = \mathbf{a}_{0:H}$ starting from the given initial state $\mathbf{s}_0$.

In the *risk-neutral* setting, the objective is to optimize the expected *return*, the expected sum of future rewards accumulated from some time instant h up to some terminal time H when starting in some initial state $\mathbf{s}_h$:

$$V_h^{\pi}(\mathbf{s}_h) := \mathbb{E}_{\mathbf{s}_{h+1:H}}\left[\sum_{t=h}^{H} r(\mathbf{s}_t, \mathbf{a}_t)\right], \tag{1}$$

where $\mathbf{a}_t = \pi_t(\mathbf{s}_t)$ for every $t = h, h+1, \ldots H$. The optimal policy π^* of a *risk-neutral* agent attains the maximum in (1) starting from state $\mathbf{s}_0$ at time $h = 0$, in which case we write $V_0^*(\mathbf{s}_0) = V_0^{\pi^*}(\mathbf{s}_0)$. Furthermore, the optimal value function at every time instant h satisfies the *Bellman equation*

$$V_h^*(\mathbf{s}_h) = \max_{\mathbf{a}_h \in \mathcal{A}} \mathbb{E}_{\mathbf{s}_{h+1}}\left[r(\mathbf{s}_h, \mathbf{a}_h) + V_{h+1}^*(\mathbf{s}_{h+1})\right], \tag{2}$$

which allows both π^* and V_0^* to be computed iteratively through time (Puterman 2014).

2.2 Entropic Risk-Sensitive MDPs

Risk sensitivity can be incorporated into the agent's decision making by replacing the expectation operator $\mathbb{E}_{\mathbf{s}_{h+1:H}}[\cdot]$ with a non-linear utility function. In this paper, we consider the *entropic utility*, which for $\beta \in \mathbb{R}$ and a random variable X is defined as

$$\mathcal{U}(X) := \frac{1}{\beta} \log \mathbb{E}\left[e^{\beta X}\right]. \tag{3}$$

Taylor expansion of (3) obtains the mean-variance approximation

$$\mathcal{U}(X) = \mathbb{E}[X] + \frac{\beta}{2}\mathrm{Var}[X] + O(\beta^2). \tag{4}$$

Now, interpreting variance as risk, β can be interpreted as the overall level of risk aversion of the agent: $\beta = 0$ induces *risk-neutral* behavior, while choosing $\beta > 0$ ($\beta < 0$) induces *risk-seeking* (*risk-averse*) behaviors. Thus for a risk-averse agent, the entropic utility can provide protection against high return variability.

Generalizing (1) to the risk-aware setting, define the *utility* of a policy π from time h as

$$U_h^\pi(\mathbf{s}_h) := \mathcal{U}_{\mathbf{s}_{h+1:H}}\left(\sum_{t=h}^{H} r(\mathbf{s}_t, \mathbf{a}_t)\right), \tag{5}$$

where it is understood that the expectations are computed w.r.t. the distribution of $\mathbf{s}_{h+1:H}$. Furthermore, due to the *recursive property* of entropic utility (Osogami 2012; Dowson, Morton, and Pagnoncelli 2020), the optimal utility satisfies the Bellman equation

$$U_h^*(\mathbf{s}_h) = \max_{\mathbf{a}_h \in \mathcal{A}} \mathcal{U}_{\mathbf{s}_{h+1}}\left(r(\mathbf{s}_h, \mathbf{a}_h) + U_{h+1}^*(\mathbf{s}_{h+1})\right). \tag{6}$$

The entropic utility satisfies a number of other important properties, including monotonicity, translation invariance and convexity, that readily extend to the MDP setting. We summarize the key properties used in this work below.

Lemma 1. *For any random variables X, Y such that $\mathbb{P}(X \geq Y) = 1$, $\mathcal{U}(X) \geq \mathcal{U}(Y)$. Furthermore, if c is deterministic, then $\mathcal{U}(X + c) = \mathcal{U}(X) + c$.*

For detailed discussion and proofs of these and other properties, we refer the reader to Föllmer and Schied (2002); Maccheroni, Marinacci, and Rustichini (2006). Furthermore, the entropic utility is the *only* class of utility that satisfies these properties and for which (5) is generally equal to (6) (Kupper and Schachermayer 2009), making it a suitable candidate for incorporating risk-awareness into end-to-end planning.

3 Risk-Sensitive Planning

The goal of planning is to avoid the expensive computation of (2) by optimizing for the sequence of actions directly. In this section, we formally define our proposed method, RAPTOR, that computes a plan for the risk-sensitive analogue (6). To this end, we employ stochastic computation graphs and the reparameterization trick for the entropic utility.

3.1 Stochastic Computation Graphs

A *stochastic computation graph* (Schulman et al. 2015) is a representation of a model that mixes deterministic computation with random variables drawn from distributions whose parameters depend on the results of previous computations. Its purpose is to define all dependencies between the variables of an acyclic generative model in order to allow the development of efficient sampling and gradient estimators.

Formally, a stochastic computation graph $\mathcal{G} = (V, E)$ is a directed acyclic graph having three disjoint sets of nodes: (i) input nodes Θ that are directly observable (including data or model parameters with respect to which we differentiate); (ii) deterministic nodes D that define functions of their parent nodes; and (iii) stochastic nodes S corresponding to random variables, whose conditional distributions are specified by functions with parameters depending on their parent nodes. For each edge $(u, v) \in E$, v is a child node whose value or probability distribution depends on its parent node u. The notation $v \prec w$ denotes that there exists a dependency path from node v to node w in $\mathcal{G}$. If the path only traverses deterministic nodes, then we annotate the path with the superscript D ($v \prec^D w$), whereas a path that is subject to stochasticity — due to traversing at least one stochastic node — is denoted as $v \prec^S w$.

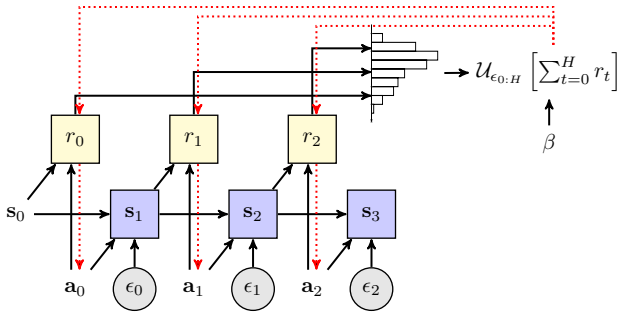

Figure 2: The stochastic computation graph of RAPTOR for three decision steps. Following Schulman et al. (2015), square and rounded nodes show deterministic and stochastic nodes, respectively. The input nodes are drawn without borders. Note that all state nodes $\mathbf{s}_{t+1}$ ($t = 0, \dots$) are deterministic due to the reparameterization via ϵ_t. During the forward pass, the inputs of the model along with a batch of samples of ϵ_t induce an empirical distribution (histogram) over $\sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t)$. From this, we compute the risk-sensitive objective function $\mathcal{U}$, which is a symbolic function of the samples and their sufficient statistics. We show the flow of gradients during backpropagation in red dotted lines.

In a stochastic computation graph, leaf nodes $\mathbf{r}$ jointly define the symbolic objective function: $J(\theta) = \mathcal{U}_{p_\theta}(g(\mathbf{r}))$. Here, $J(\theta)$ takes as input the nodes θ, and returns the utility of a differentiable real-valued function g w.r.t. the probability distribution p_θ induced by the computation graph. In a CSA-MDP, the leaf nodes $\mathbf{r} = \{r_t\}_{t=0}^{H}$ represent the immediate rewards collected at each time instant, and $g(\mathbf{r}) = \sum_t r_t$. Crucially, when there exists a path in the graph such that $\theta \prec^S J(\theta)$, then $J(\theta)$ can no longer be optimized directly in an end-to-end manner via automatic differentiation (Griewank and Walther 2008), because the utility cannot be computed in closed-form. To get around this difficulty, we introduce the reparameterization trick for location-scale families in the next subsection.

The model defined by RAPTOR naturally lends itself to a stochastic computation graph, shown in Figure 2. Here, the set of input nodes Θ includes the initial state $\mathbf{s}_0$, actions $\mathbf{a}_0, \dots \mathbf{a}_H$, and the risk aversion parameter β (3). The stochastic transition $\mathbf{s}_{t+1} = \phi(\mathbf{s}_t, \mathbf{a}_t, \epsilon_t)$ can be encoded with three edges: $\mathbf{s}_t \to \mathbf{s}_{t+1}$, $\mathbf{a}_t \to \mathbf{s}_{t+1}$, and $\epsilon_t \to \mathbf{s}_{t+1}$, where ϵ_t is an i.i.d. random noise and $\phi(\cdot)$ is a function differentiable w.r.t. actions. Similarly, the reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ is described by two edges: $\mathbf{s}_t \to r_t$ and $\mathbf{a}_t \to r_t$. From each r_t node, an edge extends to the objective node $\mathcal{U}_{\epsilon_{0:H}}\left(\sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t)\right)$. These edge dependencies in Figure 2 are drawn as solid black arrows.

3.2 Risk-Sensitive Planning by Backpropagation

The difficulty of our approach lies in computing the gradients of $J(\theta)$. To perform this computation, the probabil-

ity density *reparameterization trick* is widely used (Kingma and Welling 2014; Blundell et al. 2015; Figurnov, Mohamed, and Mnih 2018; Schulman et al. 2015).

Suppose there is a stochastic node $\mathbf{y} \sim p(\cdot|\text{parents}(\mathbf{y}))$ in a stochastic computation graph $\mathcal{G}$, whose distribution depends on the input nodes θ (implicitly) and the parent nodes of $\mathbf{y}$. The reparameterization trick transforms $\mathbf{y}$—which *blocks* the gradient of the objective to propagate backwards to $\text{parent}(\mathbf{y})$ in the graph—into a deterministic differentiable function $\phi(\theta, \epsilon)$[1], where $\epsilon \sim p(\epsilon)$ is an independent random noise. Assuming that $v \prec^D \mathbf{y}$ for all ancestral nodes v of $\mathbf{y}$, samples of $\mathbf{y}$ can be obtained by sampling ϵ and applying the transformation $\phi(\theta, \epsilon)$. Hence, by transforming all stochastic nodes in $\mathcal{G}$ into deterministic nodes in this way, we can efficiently forward-sample the objective $J(\theta)$. Using these samples, we can then compute sufficient statistics necessary for the unbiased Monte Carlo estimation of the objective and its gradient. As a result, the gradient can now flow through the deterministic node $\phi(\theta, \epsilon)$, allowing end-to-end training of θ via backpropagation.

Specializing this idea to RAPTOR, note that $\mathbf{a}_t \prec^S \mathcal{U}_\epsilon(\cdot)$ holds for all $t = 0, \ldots, H$, which prevents us from optimizing the sequence of actions via backpropagation. To circumvent this difficulty, we note the following relations:

$$\mathbf{s}_{t+1} = \phi(\mathbf{s}_t, \mathbf{a}_t, \epsilon_t) \tag{7}$$

$$\begin{aligned} r_t &= r(\mathbf{s}_t, \mathbf{a}_t) = r(\phi(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \epsilon_{t-1}), \mathbf{a}_t) \\ &= r(\phi(\phi(\phi(\mathbf{s}_0, \mathbf{a}_0, \epsilon_0), \cdots), \mathbf{a}_t) \end{aligned} \tag{8}$$

where $\phi(\cdot)$ is a deterministic differentiable (w.r.t. $\mathbf{a}_t$) function and $\epsilon_t \sim p(\epsilon_t)$ is an independent random noise. In (8), we represent the reward function at time t in a nested form using the function $\phi(\cdot)$. This allows us to reparameterize all the state nodes $\mathbf{s}_t$ starting from $t = 1$, resulting in $\mathbf{a}_t \prec^D \mathcal{U}_\epsilon(\cdot)$ for all t. Now, sampling $\boldsymbol{\epsilon} = (\epsilon_0, \ldots, \epsilon_H)$ from $p(\boldsymbol{\epsilon}) = \prod_{t=0}^{H} p(\epsilon_t)$ and applying (7) and (8) allows us to obtain samples of $\sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t)$. These samples are then used to estimate sufficient statistics, including the mean in (3) and the mean and variance in (4), and in turn the utility objective and its gradient. In practice, RAPTOR uses PyTorch (Paszke et al. 2019) to symbolically differentiate the Monte Carlo estimation of the entropic (or mean-variance) utility via automatic differentiation.

In the context of Figure 2, the flow of gradients in backpropagation is depicted using red dotted lines. As in BackpropPlan, the only set of learnable parameters in our model is the sequence of actions $\mathbf{a}_0, \ldots \mathbf{a}_H$. In other words, *we optimize for a risk-sensitive straight-line plan in an end-to-end manner via backpropagation*. Although committing to a fixed sequence of actions (i.e., straight-line plan) in a stochastic environment can be sub-optimal, we now show that the straight-line utility (denote it as $J(\mathbf{a}_{0:H})$) lower bounds the optimal utility when employing the entropic risk.

[1]For example, the location-scale family of distributions (such as Normal, Gamma, Uniform, and Cauchy) is defined as $\mathbf{y} = \phi(\theta, \epsilon) = \mu_\theta + \sigma_\theta \epsilon$, where μ_θ and σ_θ correspond to the location and scale parameters of p_θ, respectively (Bueno et al. 2019). For the exponential distribution with the rate parameter λ, we use $\phi(\theta, \epsilon) = {}^1/_\lambda \cdot \epsilon$ where $\epsilon \sim \exp(1)$.

3.3 Risk-Sensitive Anticipatory Sampling

Optimizing $J(\mathbf{a}_{0:H})$ by backpropagation avoids the computationally expensive iteration of (6) while outputs the maximizing sequence $\mathbf{a}^*_{0:H}$ of $J(\mathbf{a}_{0:H})$ as an optimal plan. The optimal utility of this plan starting in state $\mathbf{s}_0$, denoted $u_{SL}(\mathbf{s}_0) = J(\mathbf{a}^*_{0:H})$, is called the *straight-line utility*

$$u_{SL}(\mathbf{s}_0) := \max_{\mathbf{a}_{0:H}} \mathcal{U}_{\epsilon_{0:H}} \left(\sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t) \right). \tag{9}$$

In general, (9) and (6) do not have the same values. In essence, the issue here is stochasticity, which prevents the open loop policy (plan) to respond to changes in the state process that deviate significantly from their anticipated trajectories. However, as in the risk-neutral planning setting (Raghavan et al. 2017), we expect the risk-aware straight-line utility to be a lower bound to the optimal utility. We prove this for the first time in the risk-sensitive setting.

Theorem 1. *The straight-line utility $u_{SL}(\mathbf{s}_0)$ is a lower bound to the optimal utility, $U^*_0(\mathbf{s}_0) \geq u_{SL}(\mathbf{s}_0)$.*

Proof. Let ϵ be a random variable, A be an arbitrary set and $X_a = g_a(\epsilon)$ for $a \in A$ and some functions g_a. Since $\max_{a \in A} X_a \geq X_b$ for every $b \in A$ with probability one, and since $\mathcal{U}_\epsilon(\cdot)$ is monotone, it follows that $\mathcal{U}_\epsilon(\max_{a \in A} X_a) \geq \mathcal{U}_\epsilon(X_b)$. Since b is arbitrary, we have $\mathcal{U}_\epsilon(\max_{a \in A} X_a) \geq \max_{a \in A} \mathcal{U}_\epsilon(X_a)$.

Next, we write $\mathbf{s}_{h+1} = \phi(\mathbf{s}_h, \mathbf{a}_h, \epsilon_h)$ when the meaning is clear from the context. Then, using the recursive property and Lemma 1:

$$\begin{aligned} &U^*_0(\mathbf{s}_0) \\ &= \max_{\mathbf{a}_0} \mathcal{U}_{\epsilon_0}(r(\mathbf{s}_0, \mathbf{a}_0) + U^*_1(\mathbf{s}_1)) \\ &= \max_{\mathbf{a}_0} \mathcal{U}_{\epsilon_0} \left(r(\mathbf{s}_0, \mathbf{a}_0) + \max_{\mathbf{a}_1} \mathcal{U}_{\epsilon_1}(r(\mathbf{s}_1, \mathbf{a}_1) + U^*_2(\mathbf{s}_2)) \right) \\ &\geq \max_{\mathbf{a}_0} \max_{\mathbf{a}_1} \mathcal{U}_{\epsilon_0}(r(\mathbf{s}_0, \mathbf{a}_0) + \mathcal{U}_{\epsilon_1}(r(\mathbf{s}_1, \mathbf{a}_1) + U^*_2(\mathbf{s}_2))) \\ &= \max_{\mathbf{a}_0, \mathbf{a}_1} \mathcal{U}_{\epsilon_0}(\mathcal{U}_{\epsilon_1}(r(\mathbf{s}_0, \mathbf{a}_0) + r(\mathbf{s}_1, \mathbf{a}_1) + U^*_2(\mathbf{s}_2))) \\ &= \max_{\mathbf{a}_0, \mathbf{a}_1} \mathcal{U}_{\epsilon_0, \epsilon_1}(r(\mathbf{s}_0, \mathbf{a}_0) + r(\mathbf{s}_1, \mathbf{a}_1) + U^*_2(\mathbf{s}_2)) \\ &\geq \ldots \\ &= \max_{\mathbf{a}_{0:H}} \mathcal{U}_{\epsilon_{0:H}}(r(\mathbf{s}_0, \mathbf{a}_0) + \cdots + r(\mathbf{s}_H, \mathbf{a}_H)) \\ &= u_{SL}(\mathbf{s}_0). \end{aligned}$$

This completes the proof. □

4 Experiments

This section introduces two domains which are used to analyze the performance of end-to-end risk-sensitive planning by gradient descent, as well as, the performance metrics used and the resultant performance.

4.1 Navigation

The navigation domain (Faulwasser and Findeisen 2009) involves finding an optimal (shortest) path from a fixed state to a fixed region in two-dimensional space, in which

states and actions are both continuous variables. The state $\mathbf{s}_t = (s_{t,x}, s_{t,y})$ is two-dimensional and represents the location of the agent at each time t. The actions $\mathbf{a}_t = (a_{t,x}, a_{t,y})$ are also two-dimensional, and represent the coordinate displacements of the agent at each time t, constrained by

$$-2 \leq a_{t,x}, a_{t,y} \leq 2.$$

The objective of the navigation domain is to reach the goal region in a minimal number of time steps. For simplicity, the reward is computed based on the Euclidean distance from the current state $\mathbf{s}_t$ to the center of the goal region $\boldsymbol{g} = (g_x, g_y)$, at each time step:

$$r(\mathbf{s}_t, \mathbf{a}_t) = \sqrt{(g_x - s_{t,x})^2 + (g_y - s_{t,y})^2}.$$

As well, the navigation domain contains a high-variance zone where noise is added to the resultant state $\mathbf{s}_{t+1}$ based on the amount that the agent crosses this zone. In addition, for each next state trajectory not crossing the high variance zone there is a small variance applied. Thus, the transition function is

$$\phi(\mathbf{s}_t, \mathbf{a}_t, \epsilon_t) = \mathbf{s}_t + \mathbf{a}_t + 0.1 * \boldsymbol{crossing}_t * \epsilon_t + 0.01 * \epsilon_t,$$

where ϵ_t follows the standard normal distribution and $crossing_t$ is the length of the trajectory from $\mathbf{s}_t$ to $\mathbf{s}_t + \mathbf{a}_t$ that crosses the high variance zone.

4.2 Reservoir Control

Reservoir control (Yeh 1985) consists of multiple interconnected water reservoirs, between which the flow of water is controlled by dams. For each reservoir i, there exists a single state $s_i \in \mathbb{R}$ denoting its water level, and there also exists at least one connection to a downstream reservoir. An action a_{ij} corresponds to the downstream flow from reservoir i through its corresponding connection j. The actions are constrained such that the total outflow of a reservoir does not exceed the current water level of that reservoir, nor is the outflow negative (water can only flow downstream), so that $\sum_j a_{ij} \in [0, s_i]$.

The objective of an agent is to ensure that the water level in each reservoir is within a safe range $[U_i, L_i]$. Thus, the reward is calculated as follows:

$$cost_i = \begin{cases} -50(s_{t,i} - U_i), & \text{if } s_{t,i} \geq U_i \\ -0.005(L_i - s_{t,i}), & \text{if } s_{t,i} \leq L_i \\ 0, & \text{otherwise} \end{cases}$$

$$r(\mathbf{s}_t, \mathbf{a}_t) = \sum_{i \in \mathcal{S}} cost_i.$$

Note that the upper bound penalty is weighted more heavily than the lower bound penalty, since we assume overflows are more detrimental than low water levels. That is, overflows can cause flooding and significant damage to nearby communities and hence less favorable than temporary water shortages, which could be corrected fairly easily by, for example, supplementing water from secondary sources.

The amount of rainfall adds stochasticity at each time step, which is modelled at each time t as an exponentially-distributed random variable ϵ_t with rate parameter λ. Now, the transition model for reservoir control becomes:

$$\phi(\mathbf{s}_t, \mathbf{a}_t, \epsilon_t) = \mathbf{s}_t - \mathbf{a}_t A_d + \mathbf{a}_t A_u + \epsilon_t,$$

where A_d is an adjacency matrix representing connections from upstream to downstream reservoirs and A_u is an adjacency matrix representing incoming connections from each upstream reservoir:

$$A_{d,ij} = \begin{cases} 1, & \text{if there is a connection from } i \text{ to } j, \text{ or} \\ & i = j \text{ and } i \text{ has water flow out of the system} \\ 0, & \text{otherwise} \end{cases}$$

$$A_{u,ij} = \begin{cases} 1, & \text{if there is a connection from } j \text{ to } i \\ 0, & \text{otherwise} \end{cases}.$$

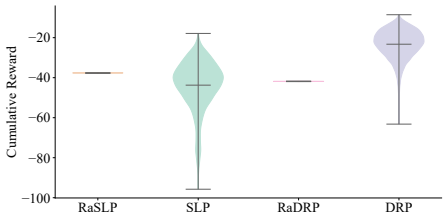

Figure 3: The cumulative reward distributions for the navigation domain showing extreme and mean values with increasing risk sensitivity from left to right. The actions were learned from the mean-variance approximation of entropic risk.

4.3 Discussion

Navigation Results In Figure 3, it can be seen that while the risk-neutral plan (i.e., $\beta = 0$) has a higher expected cumulative reward, the risk-sensitive plan leads to significantly smaller variance in cumulative reward. This makes sense intuitively, since in Figure 1, we see that the risk-neutral actions—cutting directly through the high variance zone—cause the earlier actions to be much closer to the goal region, thereby increasing the overall reward. This, however, comes at the expense of large variance and causes many of the risk-neutral trajectories to fail in reaching the goal region, as corroborated in Table 1. This high variability in both return and state trajectories is mitigated by RAPTOR, which leads to more predictable behavior.

β	Misses (%)
0.0	92.29
−1.25	0.17
−2.5	0.09

Table 1: The percent of trajectory terminations outside of the goal region for each β parameter. Results were gathered by applying trained action sequences 300,000 times.

Reservoir Results In Figure 4, it can be seen that expected cumulative reward trends higher as risk-sensitivity increases, while the opposite is true for cumulative reward

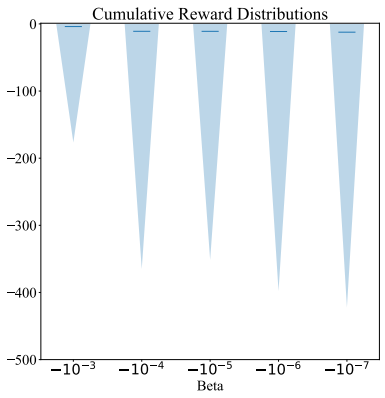

Figure 4: The cumulative reward distributions for the reservoir domain with the horizontal lines indicating mean. Distributions are generated from the 275,000 applications of trained action sequences with risk-sensitivity decreasing from left to right. The actions were learned directly through the entropic risk measure.

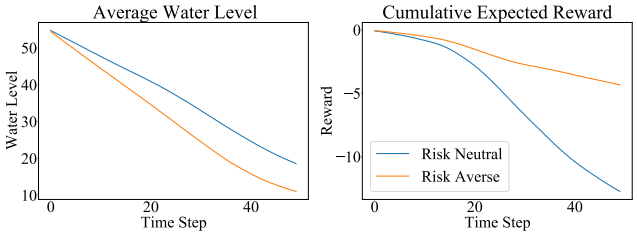

Figure 5: These results were gathered with 275,000 applications of the learned action sets. (left) represents the mean water level for the reservoir system at each time step in the horizon, averaged over the 275,000 applications. (right) represents the accumulated reward up to each time step, averaged over the 275,000 applications.

β	Overflows (%)
-10^{-7}	0.67
-10^{-6}	0.65
-10^{-5}	0.54
-10^{-4}	0.61
-10^{-3}	0.17

Table 2: The average rate of overflows per time step, calculated by applying actions learned for each β 275,000 times.

variance. An intuitive explanation for this result is that, as shown in Figure 5 and Figure 6, RAPTOR preemptively sets the water level lower thereby increasing the expected reward. Moreover, since the reward is zero when within the acceptable water level range, variance only increases when the water level is outside these bounds. Therefore, as the risk-sensitive actions have less overflows (Table 2), and overflows are the larger contributor to penalties, it is reasonable that RAPTOR would have lower variance as well.

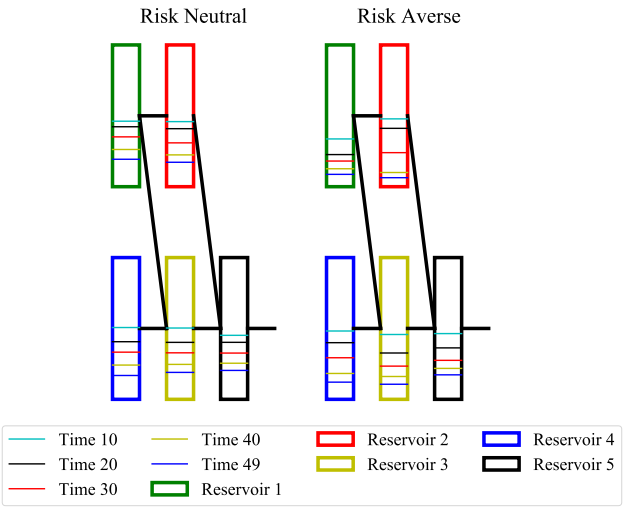

Figure 6: The risk-neutral (left) and risk-sensitive actions (right) were applied over 275,000 time horizons and the average water level at the corresponding time steps were plotted. As can be seen, on average the risk-sensitive action set preemptively keeps the water level lower to avoid overflows.

5 Conclusion and Future Work

In this paper, we have proposed RAPTOR, a scalable end-to-end risk-aware planner based on gradient descent on a risk-sensitive utility function. To this end, we have extended BackpropPlan in order to accommodate stochastic transitions, by representing the planning problem as a stochastic computation graph and applying the reparameterization trick. Critically, we have introduced the entropic utility as objective function, which can be seamlessly embedded in the graph as a symbolic objective. Then, *risk-aware planning can be done end-to-end* using an off-the-shelf automatic differentiation tool (e.g., PyTorch) by exploiting samples of the utility and their sufficient statistics. Although the resulting plan optimizes a straight-line utility, we have shown that this utility lower bounds the optimal entropic utility. The effectiveness of our approach has been demonstrated through experiments on two highly stochastic domains: Navigation and Reservoir.

As future work, we intend to consider scalable closed-loop policies by following the deep reactive policy approach (Bueno et al. 2019). Additionally, we note that extending our current approach to handle hybrid (mixed continuous and discrete) MDPs should be straightforward by using tricks such as projected gradients. A final extension of our work could investigate planning using other symbolic utility functions in place of the entropic utility, such as CVaR (Chow et al. 2015).

References

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight Uncertainty in Neural Network. In *ICML*, volume 37 of *PMLR*, 1613–1622. PMLR.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-

Theoretic Planning: Structural Assumptions and Computational Leverage. *J. Artif. Int. Res.* 11(1): 1–94. ISSN 1076-9757.

Bueno, T. P.; de Barros, L. N.; Mauá, D. D.; and Sanner, S. 2019. Deep reactive policies for planning in stochastic nonlinear domains. In *AAAI*, volume 33, 7530–7537.

Chow, Y.; Tamar, A.; Mannor, S.; and Pavone, M. 2015. Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In *NeurIPS*.

Defourny, B.; Ernst, D.; and Wehenkel, L. 2008. Risk-aware decision making and dynamic programming .

Dowson, O.; Morton, D. P.; and Pagnoncelli, B. K. 2020. Multistage stochastic programs with the entropic risk measure.

Faria, J. 2018. Machine Learning Safety: An Overview. *Safety-critical Systems Symposium 2018 (SSS'18)* .

Faulwasser, T.; and Findeisen, R. 2009. Nonlinear Model Predictive Path-Following Control. *Nonlinear Model Predictive Control - Towards New Challenging Applications* 335–343. URL http://infoscience.epfl.ch/record/184946.

Figurnov, M.; Mohamed, S.; and Mnih, A. 2018. Implicit Reparameterization Gradients. In *NeurIPS*, volume 31.

Föllmer, H.; and Schied, A. 2002. Convex measures of risk and trading constraints. *Finance and stochastics* 6(4): 429–447.

Griewank, A.; and Walther, A. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. USA: Society for Industrial and Applied Mathematics, second edition. ISBN 0898716594.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

Kupper, M.; and Schachermayer, W. 2009. Representation results for law invariant time consistent functions. *Mathematics and Financial Economics* 2(3): 189–210.

Maccheroni, F.; Marinacci, M.; and Rustichini, A. 2006. Ambiguity aversion, robustness, and the variational representation of preferences. *Econometrica* 74(6): 1447–1498.

Mannor, S.; and Tsitsiklis, J. N. 2011. Mean-variance optimization in Markov decision processes. In *ICML*, 177–184.

Moldovan, T. M. 2014. *Safety, risk awareness and exploration in reinforcement learning*. Ph.D. thesis, University of California, Berkeley.

Osogami, T. 2012. Robustness and risk-sensitivity in Markov decision processes. *NeurIPS* 25: 233–241.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 8024–8035.

Pereira, A.; and Thomas, C. 2020. Challenges of Machine Learning Applied to Safety-Critical Cyber-Physical Systems. *Machine Learning and Knowledge Extraction* 2. doi:10.3390/make2040031.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Raghavan, A.; Sanner, S.; Khardon, R.; Tadepalli, P.; and Fern, A. 2017. Hindsight optimization for hybrid state and action MDPs. In *AAAI*, volume 31.

Ruszczyński, A. 2010. Risk-averse dynamic programming for Markov decision processes. *Mathematical programming* 125(2): 235–261.

Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015. Gradient Estimation Using Stochastic Computation Graphs. In *NeurIPS*, 3528–3536.

Tieleman, T.; and Hinton, G. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Wu, G.; Say, B.; and Sanner, S. 2017. Scalable Planning with Tensorflow for Hybrid Nonlinear Domains. In *NeurIPS*.

Yeh, W. W.-G. 1985. Reservoir management and operations models: A state-of-the-art review. *Water resources research* 21(12): 1797–1818.