A Distributional Framework for Risk-Sensitive End-to-End Planning in Continuous MDPs

Noah Patton*1, Jihwan Jeong*1, Michael Gimelfarb*2, Scott Sanner2

1Department of Mechanical and Industrial Engineering, University of Toronto, 5 King’s College Rd, Toronto, ON M5S 3G8
2Vector Institute, Toronto, Canada.

noah.patton@mail.utoronto.ca, jhjeong@mie.utoronto.ca, mike.gimelfarb@mail.utoronto.ca, ssanner@mie.utoronto.ca

Abstract

Recent advances in efficient planning in deterministic or stochastic high-dimensional domains with continuous action spaces leverage backpropagation through a model of the environment to directly optimize action sequences. However, existing methods typically do not take risk into account when optimizing in stochastic domains, which can be incorporated efficiently in MDPs by optimizing a nonlinear utility function of the return distribution. We bridge this gap by introducing Risk-Aware Planning using PyTorch (RAPTOR), a novel unified framework for risk-sensitive planning through end-to-end optimization of commonly-studied risk-sensitive utility functions such as entropic utility, mean-variance optimization and CVaR. A key technical difficulty of our approach is that direct optimization of general risk-sensitive utility functions by backpropagation is impossible due to the presence of environment stochasticity. The novelty of RAPTOR lies in leveraging reparameterization of the state distribution, leading to a unique distributional perspective of end-to-end planning where the return distribution is utilized for sampling as well as optimizing risk-aware objectives by backpropagation in a unified framework. We evaluate and compare RAPTOR on three highly stochastic MDPs, including nonlinear navigation, HVAC control, and linear reservoir control, demonstrating the ability of RAPTOR to manage risk in complex continuous domains according to different notions of risk-sensitive utility.

Introduction

As machine learning models are more frequently deployed in the real world, the concern over ensuring their safety is ever-increasing (Faria 2018; Pereira and Thomas 2020). In sequential stochastic decision-making problems, 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, can be underrepresented (Moldovan 2014). The risk-averse MDP framework addresses this problem by optimizing utility functions or risk measures with favorable mathematical properties (Ruszczynski 2010).

On the other hand, planning optimizes decisions or actions given a mathematical description of the environment, thus minimizing the need to do dangerous exploration in the real world. However, despite recent advances in scalable end-to-end planning, existing approaches do not take risk into account. For instance, BackpropPlan (Wu, Say, and Sanner 2017) utilizes recent advances in deep learning and is highly scalable in continuous-state-action MDPs (CSA-MDPs). Its main limitation is that it cannot be applied to stochastic models, which could be addressed by learning reactive policies (Bueno et al. 2019). However, existing approaches do not take the distribution of the return into account and only optimize for risk-neutral policies that maximize expected return.

Utility theory offers a powerful way of modeling and evaluating preferences towards payoffs or returns of heterogeneous decision-makers (Stigler 1950; Fishburn 1970), by evaluating risk as a function of the return distribution. A variety of utility functions and risk measures have been proposed to measure risk in MDPs, including exponential/entropic utilities (Osogami 2012; Bäuerle and Rieder 2014), mean-variance approaches (Tamar, Di Castro, and Mannor 2012, 2016) and conditional value-at-risk (CVaR) (Chow et al. 2015). However, existing utility functions have different mathematical properties and assumptions, making them suitable for measuring different types of exogenous risk. For example, mean-variance approaches and entropic utilities are particularly well-suited for problems with high variance, while CVaR is more suitable for assessing tail risk. This makes a unified treatment of different utility functions particularly advantageous.

End-to-end planning can be particularly advantageous since it does not rely on the Bellman principle that can present computational challenges when considering general risk-sensitive utility functions (Defourny, Ernst, and Wehenkel 2008; Mannor and Tsitsiklis 2011).

To this end, we propose RAPTOR (Risk-Aware Planning using PyTorch) for scalable risk-aware end-to-end planning in CSA-MDPs by backpropagation through a general differentiable utility function. 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 (Figure 2). While we cannot directly formalize a distributionally-defined utility function in closed-form for end-to-end planning, for many classes of problems we can reparameterize the objective and apply stochastic backpropagation, allowing us to leverage automatic differentiation tools (Paszke et al. 2019) for gradient-based optimization. This leads to a compact yet
We then leverage this computational framework to propose CSA-MDPs, defined as tuples factored (Boutilier, Dean, and Hanks 1999), such that the state is a bounded differentiable reward function, \( r: S \times A \times S \rightarrow [0, \infty) \) describes the nonlinear dynamics of the system, and \( s_0 \) is the initial state. Note that CSA-MDPs are naturally factored (Boutilier, Dean, and Hanks 1999), such that the components of \( s_{t+1} \) are mutually independent given the current state \( s_t \) and action \( a_t \).

In the risk-neutral setting, the objective function is to maximize the expected cumulative reward,

\[
V(a_0: H) := \mathbb{E}_{s_1: H \sim p} \left[ \sum_{t=0}^{H} r(s_t, a_t) \right],
\]

where state trajectories \( s_1: H = s_1, s_2, \ldots, s_H \) are sequentially forward-sampled according to \( p \), and actions can either be computed from a closed-loop policy \( \pi: S \rightarrow A \) or an open-loop plan \( a_0: H = a_0, a_1, \ldots, a_H \).

**Risk-Awareness in CSA-MDPs**

In the risk-aware setting, the expectation operator \( \mathbb{E}_{s_1: H} \) in (1) is typically replaced by a real-valued utility function \( U(\cdot) \) of the return random variable \( Z(a_0: H) := \sum_{t=0}^{H} r(s_t, a_t) \).

The general risk-averse planning problem requires finding a sequence of actions \( a_0: H \) (either open- or closed-loop) that maximizes the cumulative utility from a given initial state \( s_0 \):

\[
U(a_0: H) := \mathcal{U}_{s_1: H \sim p}(Z(a_0: H)),
\]

where it is understood that the state trajectories \( s_1: H \) evolve under \( p \) and are thus also dependent on \( a_0: H \).

When the utility functions are composed of differentiable mathematical operators, we show how they can provide an extension of end-to-end planning methods (Wu, Say, and Sanner 2017; Bueno et al. 2019) for handling environment risk in an intelligent and computationally efficient manner.

**Distributional Risk-Aware Planning**

We first provide a straightforward motivating example that illustrates the difficulties of optimizing (2) symbolically, and foreshadows the general approach we take to overcome them in general stochastic CSA-MDPs.
Example 1. Consider a simple stochastic navigation domain with 2-D continuous state \( s_i = (s_{i,x}, s_{i,y}) \) and continuous 2-D impulse \( a_i = (a_{i,x}, a_{i,y}) \), where \( s_{i+1} \sim \mathcal{N}(s_i + a_i, \sigma^2 \mathbf{I}) \) for some constant \( \sigma^2 > 0 \). The goal is to get as close as possible to a fixed position \( g = (g_x, g_y) \) in space, so the reward at time \( t \) can be chosen as \( r(s_t, a_t) = -\|s_t - g\|^2 \). For a single-stage problem, the random variable can now be concretely written as \( Z(s_0) = -\|s_0 - g\|^2 - \|s_1 - g\|^2 \), and (2) becomes

\[
\mathcal{U}_{s_1 \sim \mathcal{N}(s_0 + a_0, \sigma^2 \mathbf{I})}(-\|s_0 - g\|^2 - \|s_1 - g\|^2).
\]

However, the utility is difficult to optimize directly, but making use of the natural reparameterization of world dynamics

\[
s_{i+1} = s_i + a_i + \sigma \xi_i, \quad \xi_i \sim \mathcal{N}(0, \mathbf{I}_{2 \times 2}),
\]

the utility can now be easily optimized

\[
\mathcal{U}_{\xi_0 \sim \mathcal{N}(0, \mathbf{I})}(-\|s_0 - g\|^2 - \|s_0 + a_0 + \sigma \xi_0 - g\|^2).
\]

More concretely, we can simulate from \( Z(a_0) \) for any \( a_0 \) by sampling \( \xi^{(1)}, \ldots, \xi^{(N)} \) and computing

\[
\hat{v}^{(i)}(a_0) = -\|s_0 - g\|^2 - \|s_0 + a_0 + \sigma \xi^{(i)}_0 - g\|^2
\]

for \( i = 1, \ldots, N \), whose gradients with respect to \( a_0 \) are now trivial to compute. More importantly, a variety of risk-informed statistics can be computed by aggregating \( \hat{v}^{(1)}, \ldots, \hat{v}^{(N)} \), such as variances and quantiles, forming an empirical estimate of the utility \( \bar{U}(a_0) \approx U(a_0) \) and then optimizing it by gradient descent.

Risk-Aware Planning by Backpropagation

Based on the previous example, there are two fundamental challenges to overcome in establishing risk-aware planning with (2) in general CSA-MDPs. The first problem is how to generate samples \( \hat{v}^{(i)} \) from the return distribution \( Z(a_0) \) that facilitate back-propagation through action sequences in stochastic domains. The second problem is how to use this empirical return distribution to incorporate differentiable risk-sensitive objectives \( \bar{U}(a_0) \) for end-to-end planning.

Utility Functions as Differentiable Graphs Since (2) cannot be computed exactly for nontrivial domains and arbitrary utility functions, we focus on utility functions that can be approximated using sample estimators \( \bar{U}(a_0) \approx U(a_0) \) based on finite sets of sampled returns. Specifically, we consider estimators of the form \( \bar{U}(a_0) = g(\hat{v}^{(1)}, \ldots, \hat{v}^{(N)}) \), where \( \hat{v}^{(1)}, \ldots, \hat{v}^{(N)} \sim Z(a_0) \) is a finite set of i.i.d. samples from the return distribution and \( g \) is a differentiable aggregation function. We require that \( g \) be asymptotically unbiased, e.g. \( \mathbb{E}[g(\hat{v}^{(1)}, \ldots, \hat{v}^{(N)})] \rightarrow U(a_0) \) as \( N \rightarrow \infty \), so that the bias becomes negligible for large \( N \). Furthermore, it is also desirable that \( g \) be exchangeable, since obtaining the same return samples in a different order should not change the overall utility. We shall see later that these assumptions cover a wide variety of commonly-studied utility functions, while being compatible with end-to-end planners.

Reparameterization Trick for CSA-MDPs The first problem is how to obtain samples \( \hat{v}^{(i)}(a_0) \sim Z(a_0) \) whose gradients can be computed with respect to the action sequence. The difficulty here lies in the fact that the state \( s_0 \) is stochastic by nature — depends on the previous actions through \( p \). Thus, the gradient of \( \hat{v}^{(i)}(a_0) \) is not well-defined since \( \hat{v}^{(i)} \) is a random variable. Instead, we employ the widely-used reparameterization trick (Kingma and Welling 2014; Blundell et al. 2015; Schulman et al. 2015; Figurnov, Mohamed, and Mnih 2018), by rewriting a sample from the distribution \( s_{i+1} \sim p(\cdot | s_i, a_i) \) as the output of a deterministic differentiable function \( \phi: S \times A \times \Xi \rightarrow S: \)

\[
s_{i+1} = \phi(s_i, a_i, \xi), \quad \xi_0, \xi_1, \ldots, \xi_{H-1} \sim f_{\xi},
\]

where \( \xi_{0,H-1} = \xi_0, \xi_1, \ldots, \xi_{H-1} \) is a sequence of i.i.d. disturbances drawn from some distribution \( f_{\xi} \) on \( \Xi \), which we call a scenario. For practical illustration, if \( p(\cdot | s_i, a_i) \) belongs to a location-scale family, then \( \phi(s_i, a_i, \xi) = m(s_i, a_i) + \xi \circ v(s_i, a_i) \) for some differentiable \( \mathbb{R}^n \)-valued functions \( m \) and \( v \). In fact, the 2-D navigation domain discussed earlier is a special case with normally-distributed dynamics, but our approach extends beyond location-scale families as well (Ruiz, Titsias, and Blei 2016). Now (3) allows us to rewrite the state transition model explicitly as:

\[
s_t = \phi(s_{t-1}, a_{t-1}, \xi_{t-1}) = \phi(\phi(s_{t-2}, a_{t-2}, \xi_{t-2}), a_{t-1}, \xi_{t-1}) = \ldots = \Phi_h(s_0, a_{0:t-1}, \xi_{0:t-1}).
\]

A Distributional Perspective on End-to-End Planning

The distribution of the return \( Z(a_0) = \sum_{t=0}^H r(s_t, a_t) \) can now be reparameterized using (4), leading to

\[
Z(\xi(a_0)) = \sum_{t=0}^H r(\Phi_t(s_0, a_{0:t-1}, \xi_{0:t-1}), a_t).
\]

Since the reparameterization is exact, \( Z(\xi(a_0)) \) remains an exact distribution of the return since \( Z(a_0) = Z(\xi(a_0)) \) in distribution, but it can now be written explicitly as a trainable function of the noise generator \( f_{\xi} \). This provides a unique distributional perspective (Bellemare, Dabney, and Munos 2017) on planning. Furthermore, since \( \phi \) and \( \xi \) are differentiable in \( a_i \), the gradient of \( \hat{v}^{(i)} \sim Z(\xi(a_0)) \) is easily computable by forward-sampling \( g(\hat{v}^{(i)}, \ldots, \hat{v}^{(N)}) \) and backpropagating. Namely, given \( U(a_0) \approx g(\hat{v}^{(1)}, \ldots, \hat{v}^{(N)}) \), we have:

\[
\frac{\partial U(a_0)}{\partial a_i} = \frac{\partial}{\partial a_i} U_{s_0 \sim p(Z(a_0); \xi)} = \frac{\partial}{\partial a_i} U_{s_0 \sim f_{\xi}(Z(\xi(a_0))); \xi} \approx \frac{\partial}{\partial a_i} U_{s_0 \sim f_{\xi}(Z(\xi(a_0))); \xi} = \sum_{i=1}^N \frac{\partial g}{\partial \hat{v}^{(i)}} \frac{\partial \hat{v}^{(i)}}{\partial a_i} = \sum_{i=1}^N w_i \frac{\partial \hat{v}^{(i)}}{\partial a_i}.
\]

Based on (6), we can now interpret utility functions as (non-uniform) weightings \( w_i \) of the gradients of the return samples.

\footnote{For \( t = 0 \) we simply define \( \Psi_0(s_0, a_{0:t-1}, \xi_{0:t-1}) := s_0 \) to be consistent with the derivations that follow. We also do the same for \( \Psi_0 \) defined in the later section.}
We now show how the distributional perspective (5), together with judicious choices for $g$, allows for scalable risk-aware planning using a variety of utilities.

Risk-Aware Planning with Entropic Utility  The entropic utility is defined as $U(X; \beta) = -\frac{1}{\beta} \log \mathbb{E}[e^{-\beta X}]$, where $\beta \in \mathbb{R}$ is a fixed parameter that controls the level of risk aversion (Bäuerle and Rieder 2014). Applying (5):

$$
\frac{\partial}{\partial a_t} U_{s_t \sim \nu}(Z(a_0:H); \beta) \propto \frac{\partial}{\partial a_t} \mathbb{E}_{s_t \sim \nu}[e^{-\beta Z(a_0:H)}]
$$

This corresponds to an estimator $g : (v^{(1)}, \ldots, v^{(N)}) \mapsto -\frac{1}{\beta} \log \frac{1}{N} \sum_{i=1}^{N} e^{-\beta v^{(i)}}$. Referring back to (6), $w_i \propto e^{-\beta v^{(i)}}$ assigns more importance to negative returns for $\beta < 0$, and less for $\beta > 0$.

Risk-Aware Planning with CVaR  CVaR is defined as the $\alpha$-percentile of the loss or return realizations. Another property of (6) is that the specific structure of $U(a_0:H)$, combined with the use of the chain rule for total derivatives, reduces to a Monte-carlo sample-based approximation of the gradient. For example, for the risk-neutral utility $g(v^{(1)}, \ldots, v^{(N)}) = \frac{1}{N} \sum_{i=1}^{N} v^{(i)}$, the sample return gradients are weighted uniformly, e.g. $w_i = \frac{1}{N}$. However, for strictly nonlinear $g$, the corresponding return gradients will not be weighted equally.

Finally, given a positive learning rate $\eta$, the risk-aware plan can be updated by gradient ascent

$$
\alpha_t = \alpha_t + \eta \sum_{i=1}^{N} w_i \frac{\partial v^{(i)}}{\partial t}, \quad t = 0, 1 \ldots H.
$$

We have now also addressed the second problem above, since the aggregation function $g$ allows the gradients of $v^{(i)}$ to flow through in an end-to-end manner. This makes our approach easy to implement using modern deep learning libraries such as PyTorch. Thus, we refer to our overall approach as RAPTOR (Risk-Aware Planning using PyTORch).
Unfortunately, VaR$_a[X]$ is not differentiable, so we apply the straight-through gradient trick (Bengio, Léonard, and Courville 2013; Courbariaux et al. 2016). Specifically, RAPTOR sorts $\hat{\phi}(1), \ldots, \hat{\phi}(N)$ to obtain order statistics $\hat{\phi}(i_1) \leq \hat{\phi}(i_2) \leq \cdots \leq \hat{\phi}(i_N)$, uses them to estimate VaR$_a$ as $\hat{\phi}(\lfloor N(1 - \alpha) \rfloor)$, and then backpropagates through the smallest $\lfloor N(1 - \alpha) \rfloor$ samples without passing the gradient through $\hat{\phi}(\lfloor N(1 - \alpha) \rfloor)$ (Kolla et al. 2019):

$$\frac{\partial}{\partial a_t} \text{CVaR}_a[Z(a_{0:t})] \approx \frac{1}{N(1 - \alpha)} \frac{\partial}{\partial a_t} \sum_{i=1}^N \left[ \hat{\phi}(i) \leq \hat{\phi}(\lfloor N(1 - \alpha) \rfloor) \right].$$

This corresponds to $g : (\hat{\phi}(1), \ldots, \hat{\phi}(N)) \mapsto \sum_{i=1}^N \left[ \hat{\phi}(i) \leq \hat{\phi}(\lfloor N(1 - \alpha) \rfloor) \right]$ with the approximated gradient above. We also have $w_t = 0$ for those samples $\hat{\phi}(i)$ that lie above VaR$_a$, and $w_t \propto 1$ is distributed equally among the rest of the samples that lie in the tail of the empirical return distribution.

Other utility functions based on differentiable statistics of the samples $\hat{\phi}(i)$, such as quadratic utility (Luenberger et al. 1997), (mean-)semideviation/semivariance (Ogryczak and Tal and Teboulle 2007) have also been studied in the literature,

Risk-Aware Planning for Closed-Loop Policies

The optimization of $U(a_{0:H})$ provides a maximizing plan $\hat{a}_{0:H}$, that commits to a fixed sequence of actions starting from $s_0$ for the entirety of the planning horizon. For this reason, we call this the risk-aware straight-line plan (SLP).

Limitations of Risk-Aware Straight-Line Planning

The limitation of an SLP is that it cannot respond to significant deviations in the state trajectory from its “anticipated” path without online replanning. This is a particular challenge in highly stochastic domains with risky states and fast computation requirements. For instance, a planner that controls a helicopter could crash if it is unable to correct for the helicopter’s accumulated fluctuations in position due to wind. On the other hand, in the case of maximizing the entropic utility objective, we can prove that $U(a_{0:H})$ provides a (potentially tight) lower bound to the maximum entropic utility of a closed-loop policy (Mercier and Van Hentenryck 2008), which has been used extensively in risk-neutral planning (Burns et al. 2012; Issakimuthu et al. 2015; Raghavan et al. 2017). A proof can be found in the Appendix.

**Theorem 1.** The entropic utility of an optimal risk-aware SLP is a lower bound of the entropic utility of the corresponding optimal risk-aware closed-loop policy.

A Distributional Perspective on Learning Closed-Loop Policies

Fortunately, it is possible to compute optimal risk-aware closed-loop policies efficiently in CSA-MDPs by learning a parameterized risk-aware deep reactive policy (DRP) $\pi_\theta : S \rightarrow A$ (Bueno et al. 2019) by maximizing

$$U(\theta) = \mathbb{E}_{s_{1:H} \sim \rho} \left( \sum_{t=0}^H r(s_t, \pi_\theta(s_t)) \right)$$

over $\theta$. The difficulty here lies in that the DRP depends on the state, that in turn depends implicitly on the previous action, that depends on the previous state, and so on. Thus, the return distribution $Z(\theta) := \sum_{t=0}^H r(s_t, \pi_\theta(s_t))$ has a complex dependence on $\theta$. However, the distributional perspective of SLPs also extends to DRPs. In particular:

$$s_t = \phi(s_{t-1}, \pi_\theta(s_{t-1}), \xi_{t-1})$$

$$= \phi(\phi(s_{t-2}, \pi_\theta(s_{t-2}), \xi_{t-1}), \pi_\theta(s_{t-2}, \pi_\theta(s_{t-2}), \xi_{t-1}), \xi_{t-1})$$

$$= \cdots = \Psi(s_0, \theta, \xi_{0:t-1}).$$

and the return distribution can now be written concisely as

$$Z(\xi(\theta)) = \sum_{i=0}^H r(\Psi(s_0, \theta, \xi_{0:t-1}), \pi_\theta(\Psi(s_0, \theta, \xi_{0:t-1}))).$$

Therefore, the gradient of the sample $\hat{\phi}(i) \sim Z(\theta)$ is now easy to compute, and the DRP can be readily optimized for all utility functions introduced in the previous section.

Based on the results in this section, we define two possible instantiations of RAPTOR. The first approach computes an optimal risk-aware straight-line plan by back-propagating through the approximated utility $\tilde{U}(a_{0:H}) = g(\hat{\phi}(1), \ldots, \hat{\phi}(N))$, which we refer to as RAPTOR-SLP. The second approach approximates an optimal deep reactive policy, which we refer to as RAPTOR-DRP. The computation graph defining both instances of RAPTOR is depicted in Figure 2, where edge dependencies are drawn as solid black arrows and gradients are drawn as dashed red arrows.

**Experiments**

This section describes three domains which are used to analyze the performance of RAPTOR-SLP and RAPTOR-DRP for entropic (ENT), mean-variance (MV), and CVaR utilities along with comparison to the risk-neutral (RN) baseline. In all experiments, we used Adam as the optimizer and selected the learning rates according to a grid search for all compared methods based on their utility (see Appendix for further experimental details). SLP does not use re-planning.

**Domain Descriptions**

**Navigation** As briefly introduced in Example 1, an agent has to find a shortest path from a fixed starting point to a fixed goal region in a two-dimensional space (Faulwasser and Frideisen 2009). While states, actions, and rewards are defined the same way as described in Example 1, we further assume that the next state transitions are subject to high variability when the agent passes through a high-variance zone in between the starting point and the goal region (see Figure 1). In particular, let $\text{crossing}_t$ denote the length of a sub-trajectory
that crosses the zone when moving from $s_t$ to $s_t + a_t$. Then, a normally distributed noise $\mathcal{N}(0, (\text{crossing}_t - \sigma_h)^2)$ is added to the next state. On the other hand, when crossing $g_t = 0$, a small noise $\mathcal{N}(0, \sigma_g^2)$ with variance $\sigma^2 \ll \sigma_t^2$ is added. Thus, the transition function can be reparameterized as:

$$
\phi(s_t, a_t, \xi_t) = s_t + a_t + \text{crossing}_t \sigma_h \xi_t + 1[\text{crossing}_t = 0] \sigma_t \xi_t, \quad \xi_t \sim \mathcal{N}(0, I_{2 \times 2}).
$$

**Reservoir Control** The reservoir control domain (Yeh 1985) involves controlling the flow of water between $N = 5$ interconnected reservoirs. The water level of each reservoir $i$ at time $t$ is represented by $s_{t,i} \in \mathbb{R}_+$. The action $a_{t,ij} = [a_t]_{ij}$ is the amount of water discharged from reservoir $i$ to $j$ for $j \in D(i)$, where $D(i)$ is the set of downstream reservoirs of $i$ (similarly, $U(i)$ is the set of upstream reservoirs). The action is constrained such that $a_{t,ik} = 0, \forall k \notin D(i)$. Also, the total outflow of a reservoir cannot exceed the reservoir’s current water level, i.e., $\sum_{j \in U(i)} a_{t,ij} \leq s_{t,i}$. The goal of the agent is to keep the water level within safe limits $[l_t, u_t]$, so we define the reward function as:

$$
r(s_t, a_t) = \sum_{i=1}^{N} R_i(t),
$$

where

$$
R_i(t) := \begin{cases} 
-P_u(s_{t,i} - u_i), & \text{if } s_{t,i} \geq u_i \\
-P_o(l_i - s_{t,i}), & \text{if } s_{t,i} \leq l_i \\
0, & \text{otherwise}
\end{cases}
$$

and where $P_u$ ($P_o$) is the penalty for water levels above (below) the upper (lower) bound. Note that $P_o$ is weighted more heavily than $P_u$, since overflows can lead to costly flooding and damages, whereas shortages can more easily be resolved by supplementing water from secondary sources. The amount of rainfall at each time step is modelled as an exponentially-distributed random variable $\xi_t$ with rate $\lambda$. Thus, switching to a component-wise view of $\phi$, for notational clarity, the transition function for each reservoir $i$ becomes

$$
\phi_i(s_t, a_t, \xi_t) = s_{t,i} - \sum_{j \in D(i)} a_{t,ij} + \sum_{j \in U(i)} a_{t,ji} + \xi_t.
$$

**HVAC Control** In the HVAC control domain, an agent modulates the volume of heated air flow into each of the $N = 5$ rooms in a house. The state consists of the temperature in each room $i$ given by $s_{t,i} \in \mathbb{R}$. The air volume, $a_t$, into each room is non-negative ($a_{t,i} \geq 0$) with constant temperature $T_c$. The objective is to keep the rooms at their set temperature and to avoid them getting too cold, so the reward function can be defined as:

$$
r(s_t, a_t) = \sum_{i=1}^{N} R_i(t),
$$

where

$$
R_i(t) := \begin{cases} 
-s_{t,i} - T_i - a_{t,i} - P_u, & \text{if } s_{t,i} \leq T_i \\
-s_{t,i} - T_i - a_{t,i}, & \text{otherwise}
\end{cases}
$$

and where $T_i$ is the target temperature for room $i$, $T_i$ is the lowest acceptable temperature, and $P_u$ is a penalty applied when the temperature drops below $T_i$. As a cost-saving measure, $T_i$ is set near $T_1$ to reduce heating expenses. Stochasticity is added in three ways for HVAC: (1) outdoor temperature is modeled as $\mathcal{N}(T_o, \sigma_o^2)$; (2) the volume of air applied to each room is $\mathcal{N}(0, \sigma_a^2)$; and (3) the heat dispersion between rooms $i$ and $j$ is also $\mathcal{N}(0, \sigma_h^2)$. Combining this, the reparameterized dynamics for room $i$ are

$$
\phi_i(s_t, a_t, \xi_t) = s_{t,i} + a_{t,i}(T_c - s_{t,i}) + \sigma_h \xi_t + \sum_{j \in A(i)} \frac{(s_{t,j} - s_{t,i})^2}{\rho_{ij}} + \sigma_h \xi_t + T_o + \sigma_h \xi_t,
$$

where $\xi_t \sim \mathcal{N}(0, 1)$, $\rho_{ij}$ is the thermal resistance between rooms $i$ and $j$, and $A(i)$ is the set of rooms adjacent to $i$.

**Performance of Risk-Aware Planning**

We analyze and compare the empirical performance of RAPTOR-SLP against a risk-neutral SLP planner as a baseline (Wu, Say, and Sanner 2017). We also conducted these evaluations for RAPTOR-DRP and risk-neutral DRP (with $\beta = 0$) (Bueno et al. 2019), but due to space limitations we defer most of the associated figures and detailed discussion to the Appendix. In short, RAPTOR-DRP and DRP generally show similar patterns as their SLP counterparts. However, DRPs sometimes underperform SLPs since they introduce additional training parameters and must learn to generalize across the entire state space. Overall, we observe that despite being a lower bound (Theorem 1), RAPTOR-SLP is very effective at optimizing nonlinear utility functions.

Figure 3 plots the distributions of returns obtained by RN and RAPTOR-SLP using the three different utility functions discussed previously. Specifically, we have varied the risk aversion parameters ($\beta$ for MV and ENT, $\alpha$ for CVaR) to see the effects of increasing risk-aversion. Indeed, as the magnitude of $\beta$ and $\alpha$ increases, the variances of the cumulative reward tend to shrink in all domains. Although RN often obtains higher expected cumulative reward, it does so only at the cost of highly variable returns. Also, RAPTOR is extremely efficient and can solve each of the three domains in less than 3 minutes on a consumer-grade PC.

**Navigation Results** Figure 3b-3c compare RN and RAPTOR-SLP on the Navigation domain. For RN, a large portion of the trajectories achieve greater cumulative rewards than those obtained by RAPTOR-SLP. However, the variability of the return of RN is generally high, such that more often than not the trajectories end up failing to reach the goal region (also see Figure 1a). In general, we can clearly observe the trade-off between risk and return by increasing the risk aversion parameter: smaller return variance at the expense of reductions in the expected return. Interestingly, RAPTOR-SLP with CVaR achieves higher expected returns than RN, while also reducing the variance significantly. One possible explanation is that the early stopping rule – necessary to avoid the problem of exploding gradients during optimization (Bueno et al. 2019) – is more difficult to implement for the risk-neutral agent with volatile returns, while RAPTOR mitigates by providing smoother and more stable returns.

**Reservoir Results** In Reservoir (Figure 3e-3f), we still see the similar pattern as above, i.e., reduced variances when increasing the risk aversion parameters. Specifically, when we compare the worst-case outcomes, we observe the clear benefit of incorporating risk into planning. Furthermore, the RAPTOR-SLP with CVaR is particularly adept at controlling the tail distribution and the worst-case outcomes, as can
Figure 3: The cumulative reward distributions obtained by risk-neutral (RN) and RAPTOR-SLP agents with three different risk utilities (ENT, MV, and CVaR) in Navigation (3a-3c), Reservoir (3d-3f), and HVAC (3g-3i) domains.

be expected from its definition presented earlier in the text. The evolution of water levels over time, presented in the Appendix, shows that RAPTOR-SLP maintains lower water levels than its risk-neutral counterpart in order to prevent the large negative reward incurred by unexpected overflows.

HVAC Results In Figure 3g-3i, not only the worst-case outcomes but also the variances stand out when we compare RN with RAPTOR-SLP. Similarly as in Reservoir, since the large negative reward is incurred when the temperature drops below $T_l$, RAPTOR-SLP tries to keep the room temperatures high enough at the cost of slightly larger heating expenses (cf. room temperatures in Appendix). On the other hand, RN simply ignores the variability in the temperature, so it often suffers a large penalty when temperature stochastically drops below $T_l$. For RAPTOR-SLP with CVaR, we observe that it is harder to optimize compared to the other two utilities when $\alpha$ is small, potentially because low $\alpha$ discards many samples (recall that CVaR sets $v_i = 0$ for samples $\bar{v}_i$ above VaR$_{\alpha}$).

Conclusion and Future Work

We proposed RAPTOR, a scalable end-to-end risk-aware planner based on gradient descent applied to a risk-sensitive utility function. In order to do this, we extended Backprop-Plan to accommodate stochastic transitions, by representing the planning problem as a stochastic computation graph and applying the reparameterization trick to the dynamics. This leads to a distributional perspective on differentiable end-to-end planning. We then showed that risk-aware planning using common utility functions – such as entropic utility, mean-variance objective and CVaR – can be interpreted as gradient-based planning where the gradients of the return are weighted (possibly non-uniformly) depending on the choice of utility function. We applied RAPTOR to learn a straight-line plan as well as deep reactive policies. Experiments on three highly stochastic domains — Navigation, Reservoir, and HVAC — demonstrated the ability of RAPTOR to plan and learn a meaningful range of risk-sensitive solutions in nonlinear continuous state-action MDPs.

As future work, we note that extending RAPTOR to handle hybrid (mixed continuous and discrete) MDPs will need to workarounds the non-differentiability of discrete transition models. Also, choosing the correct utility function for each task is a complex problem in sequential decision making and applying our framework to other classes of utility functions not mentioned here could be beneficial. Further afield, we could integrate model uncertainty for robust risk-aware planning under Bayesian models of misspecification.

9900
