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

Noah Patton*, Jihwan Jeong*, Michael Gimelfarb*†, Scott Sanner†

Department of Mechanical and Industrial Engineering, University of Toronto
5 King's College Rd, Toronto, ON M5S 3G8
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

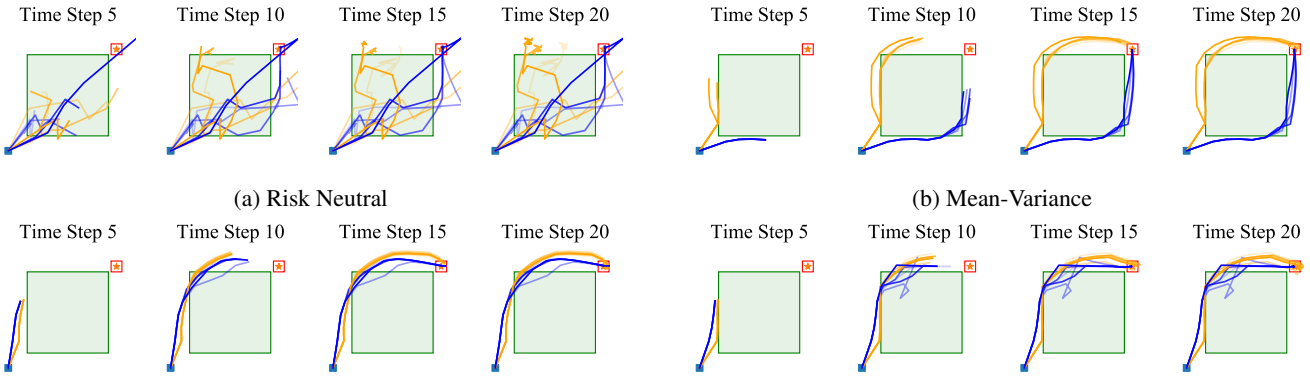
As machine learning models are more frequently deployed in the real world, the concern over ensuring their safety is ever-increasing (Faria 2018; Pereira and Thomas 2020). In sequential stochastic decision-making problems, it has been shown that optimizing the expected cumulative reward can lead to undesirable 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 (Ruszczyński 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 utilties 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)

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



(a) Risk Neutral (b) Mean-Variance

(c) Entropic (d) CVaR

Figure 1: The evolution of trajectories navigated by the agents (SLP agents are shown in orange, and DRP agents are shown in blue) in a two-dimensional Navigation domain subject to stochastic dynamics (note: multiple sample trajectories are shown simultaneously in each time step). 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 time step 20, we clearly see that the mean-variance, entropic and CVaR agents are able to get to the goal region by avoiding the box and thus failure. In contrast, the risk-neutral agents opt for the high variance but highest reward shortest path as reflected in their much noisier trajectories.

for gradient-based optimization. This leads to a compact yet exact representation of the return distribution, which allows both efficient sampling and policy/plan optimization that we refer to as the *distributional* perspective of planning. This, in turn, allows us to optimize a variety of symbolic nonlinear utility functions of the return distribution directly from forward-sampled state trajectories in a plug-and-play manner. We then leverage this computational framework to propose two approaches for optimizing *risk-aware* decisions. The first derives a *risk-sensitive straight-line plan* (SLP) that commits to a sequence of actions in advance, while the second learns a *risk-sensitive deep reactive policy* (DRP) that is particularly suited for highly stochastic domains. Both approaches integrate seamlessly into our optimization framework and can be computed efficiently in an end-to-end manner by reparameterization and backpropagation.

Indeed, as evidenced by Figure 1, RAPTOR is effective at converging toward risk-averse behaviors in highly stochastic environments in a computationally efficient manner. Overall, empirical evaluations on three highly stochastic domains involving continuous action parameters — navigation, HVAC control, and reservoir control — demonstrate that RAPTOR is a reliable and efficient end-to-end method for risk-aware planning in complex nonlinear continuous MDPs.

Preliminaries

Continuous State-Action MDPs (CSA-MDPs)

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 nonlinear 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 components of $\mathbf{s}_{t+1}$ are mutually independent given the current state $\mathbf{s}_t$ and action $\mathbf{a}_t$.

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

$$V(\mathbf{a}_{0:H}) := \mathbb{E}_{\mathbf{s}_{1:H} \sim p} \left[\sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{1}$$

where state trajectories $\mathbf{s}_{1:H} = \mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_H$ are sequentially forward-sampled according to p, and actions can either be computed from a closed-loop *policy* $\pi : \mathcal{S} \to \mathcal{A}$ or an open-loop *plan* $\mathbf{a}_{0:H} = \mathbf{a}_0, \mathbf{a}_1, \ldots \mathbf{a}_H$.

Risk-Awareness in CSA-MDPs

In the risk-aware setting, the expectation operator $\mathbb{E}_{\mathbf{s}_{1:H}}[\cdot]$ in (1) is typically replaced by a real-valued utility function $\mathcal{U}(\cdot)$ of the return random variable $Z(\mathbf{a}_{0:H}) := \sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t)$. The general risk-averse planning problem requires finding a sequence of actions $\mathbf{a}_{0:H}$ (either open- or closed-loop) that maximizes the cumulative utility from a given initial state $\mathbf{s}_0$

$$U(\mathbf{a}_{0:H}) := \mathcal{U}_{\mathbf{s}_{1:H} \sim p}(Z(\mathbf{a}_{0:H})), \tag{2}$$

where it is understood that the state trajectories $\mathbf{s}_{1:H}$ evolve under p and are thus also dependent on $\mathbf{a}_{0:H}$.

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 $\mathbf{s}_t = (s_{t,x}, s_{t,y})$ and continuous 2-D impulse $\mathbf{a}_t = (a_{t,x}, a_{t,y})$, where $\mathbf{s}_{t+1} \sim \mathcal{N}(\mathbf{s}_t + \mathbf{a}_t, \sigma^2 \mathbf{I})$ for some constant $\sigma^2 > 0$. The goal is to get as close as possible to a fixed position $\mathbf{g} = (g_x, g_y)$ in space, so the reward at time t can be chosen as $r(\mathbf{s}_t, \mathbf{a}_t) = -\|\mathbf{s}_t - \mathbf{g}\|_2$. For a single-stage problem, the return random variable can now be concretely written as $Z(\mathbf{a}_0) = -\|\mathbf{s}_0 - \mathbf{g}\|_2 - \|\mathbf{s}_1 - \mathbf{g}\|_2$, and (2) becomes

$$\mathcal{U}_{\mathbf{s}_1 \sim \mathcal{N}(\mathbf{s}_0 + \mathbf{a}_0, \sigma^2 \mathbf{I})}(-\|\mathbf{s}_0 - \mathbf{g}\|_2 - \|\mathbf{s}_1 - \mathbf{g}\|_2)\,.$$

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

$$\mathbf{s}_1 = \mathbf{s}_0 + \mathbf{a}_0 + \sigma \xi_0, \quad \xi_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{2\times 2}),$$

the utility can now be easily optimized

$$\mathcal{U}_{\xi_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}(-\|\mathbf{s}_0 - \mathbf{g}\|_2 - \|\mathbf{s}_0 + \mathbf{a}_0 + \sigma \xi_0 - \mathbf{g}\|_2)\,.$$

More concretely, we can simulate from $Z(\mathbf{a}_0)$ for any $\mathbf{a}_0$ by sampling $\xi_0^{(1)}, \ldots \xi_0^{(N)}$ and computing

$$\hat{v}^{(i)}(\mathbf{a}_0) = -\|\mathbf{s}_0 - \mathbf{g}\|_2 - \|\mathbf{s}_0 + \mathbf{a}_0 + \sigma \xi_0^{(i)} - \mathbf{g}\|_2$$

for $i = 1 \ldots N$, whose gradients with respect to $\mathbf{a}_0$ 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 $\hat{U}(\mathbf{a}_{0:H}) \approx U(\mathbf{a}_{0:H})$ 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 $Z(\mathbf{a}_{0:H})$ 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 $\hat{U}(\mathbf{a}_{0:H})$ 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 $\hat{U}(\mathbf{a}_{0:H}) \approx U(\mathbf{a}_{0:H})$ based on finite sets of sampled returns. Specifically, we consider estimators of the form $\hat{U}(\mathbf{a}_{0:H}) = g(\hat{v}^{(1)}, \ldots \hat{v}^{(N)})$, where $\hat{v}^{(1)}, \ldots \hat{v}^{(N)} \sim Z(\mathbf{a}_{0:H})$ 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. $\mathbb{E}\left[g(\hat{v}^{(1)}, \ldots \hat{v}^{(N)})\right] \to U(\mathbf{a}_{0:H})$ as $N \to \infty$, 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 $\hat{v}^{(i)}(\mathbf{a}_{0:H}) \sim Z(\mathbf{a}_{0:H})$ whose gradients can be computed with respect to the action sequence. The difficulty here lies in the fact that the state $\mathbf{s}_t$ – which is stochastic by nature – depends on the previous actions through p. Thus, the gradient of $\hat{v}^{(i)}(\mathbf{a}_{0:H})$ 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 $\mathbf{s}_{t+1} \sim p(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$ as the output of a deterministic differentiable function $\phi : \mathcal{S} \times \mathcal{A} \times \Xi \to \mathcal{S}$:

$$\mathbf{s}_{t+1} = \phi(\mathbf{s}_t, \mathbf{a}_t, \xi_t), \quad \xi_0, \xi_1, \ldots \xi_{H-1} \sim f_\xi, \tag{3}$$

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_ξ on Ξ, which we call a *scenario*. For practical illustration, if $p(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$ belongs to a location-scale family, then $\phi(\mathbf{s}_t, \mathbf{a}_t, \xi_t) = m(\mathbf{s}_t, \mathbf{a}_t) + \xi_t \odot v(\mathbf{s}_t, \mathbf{a}_t)$ 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[1]:

$$\begin{aligned}\mathbf{s}_t &= \phi(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \xi_{t-1}) \\ &= \phi(\phi(\mathbf{s}_{t-2}, \mathbf{a}_{t-2}, \xi_{t-2}), \mathbf{a}_{t-1}, \xi_{t-1}) \\ &= \ldots := \Phi_t(\mathbf{s}_0, \mathbf{a}_{0:t-1}, \xi_{0:t-1}).\end{aligned} \tag{4}$$

A Distributional Perspective on End-to-End Planning The distribution of the return $Z(\mathbf{a}_{0:H}) = \sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t)$ can now be reparameterized using (4), leading to

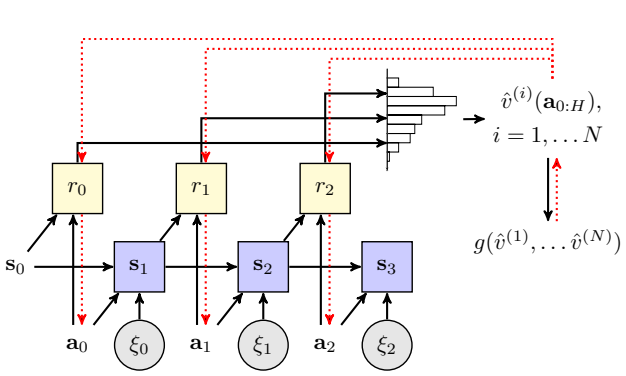
$$Z_\xi(\mathbf{a}_{0:H}) = \sum_{t=0}^{H} r(\Phi_t(\mathbf{s}_0, \mathbf{a}_{0:t-1}, \xi_{0:t-1}), \mathbf{a}_t). \tag{5}$$

Since the reparameterization is exact, $Z_\xi(\mathbf{a}_{0:H})$ remains an *exact* distribution of the return since $Z(\mathbf{a}_{0:H}) = Z_\xi(\mathbf{a}_{0:H})$ in distribution, but it can now be written explicitly as a trainable function of the noise generator f_ξ. This provides a unique *distributional* perspective (Bellemare, Dabney, and Munos 2017) on planning. Furthermore, since ϕ and r are differentiable in $\mathbf{a}_t$, the gradient of $\hat{v}^{(i)} \sim Z_\xi(\mathbf{a}_{0:H})$ is easily computable by forward-sampling $\xi_{0:H-1}^{(i)}$ and backpropagating. Namely, given $U(\mathbf{a}_{0:H}) \approx g(\hat{v}^{(1)}, \ldots \hat{v}^{(N)})$, we have:

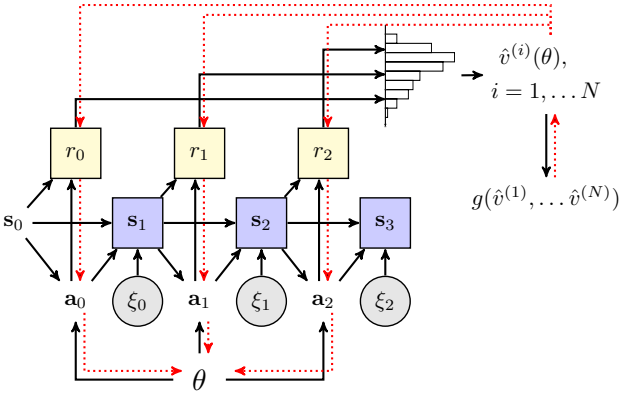
$$\begin{aligned}&\frac{\partial U(\mathbf{a}_{0:H})}{\partial \mathbf{a}_t} = \frac{\partial}{\partial \mathbf{a}_t} \mathcal{U}_{\mathbf{s}_{1:H} \sim p}(Z(\mathbf{a}_{0:H})) \\ &= \frac{\partial}{\partial \mathbf{a}_t} \mathcal{U}_{\xi_{0:H-1} \sim f_\xi}(Z_\xi(\mathbf{a}_{0:H})) \approx \frac{\partial}{\partial \mathbf{a}_t} g(\hat{v}^{(1)}, \ldots \hat{v}^{(N)}) \\ &= \sum_{i=1}^{N} \frac{\partial g}{\partial \hat{v}^{(i)}} \frac{\partial \hat{v}^{(i)}}{\partial \mathbf{a}_t} := \sum_{i=1}^{N} w_i \frac{\partial \hat{v}^{(i)}}{\partial \mathbf{a}_t}.\end{aligned} \tag{6}$$

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

[1] For $t = 0$ we simply define $\Phi_0(\mathbf{s}_0, \mathbf{a}_{0:t-1}, \xi_{0:t-1}) := \mathbf{s}_0$ to be consistent with the derivations that follow. We also do the same for Ψ_0 defined in the later section.



(a) RAPTOR-SLP



(b) RAPTOR-DRP

Figure 2: The stochastic computation graph of RAPTOR for three decision epochs. 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 states $\mathbf{s}_1, \mathbf{s}_2 \dots$ are deterministic after reparameterization via ξ_t. During the forward pass, depicted with black arrows, the inputs of the model along with a batch of samples $\xi_t^{(1)}, \dots \xi_t^{(N)} \sim f_\xi$ induce an empirical distribution over $Z(\mathbf{a}_{0:H}) = \sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{a}_t)$. From this, we approximate the utility objective $\mathcal{U}$ by its finite-sample estimator g, which is a symbolic function of the samples $\hat{v}^{(1)}, \dots \hat{v}^{(N)} \sim Z(\mathbf{a}_{0:H})$. We then backpropagate through g, depicting the flow of gradients as red dotted lines.

$\hat{v}^{(i)}$*, where the weights are determined based on the severity of the loss or return realizations*. Another property of (6) is that the specific structure of $\hat{U}(\mathbf{a}_{0:H})$, 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 $g(\hat{v}^{(1)}, \dots \hat{v}^{(N)}) = \frac{1}{N}\sum_{i=1}^{N} \hat{v}^{(i)}$, 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 η, the risk-aware plan can be updated by gradient ascent

$$\mathbf{a}_t = \mathbf{a}_t + \eta \sum_{i=1}^{N} w_i \frac{\partial \hat{v}^{(i)}}{\partial \mathbf{a}_t}, \quad t = 0, 1 \dots H.$$

We have now also addressed the second problem above, since the aggregation function g allows the gradients of $\hat{v}^{(i)}$ 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 (**R**isk-**A**ware **P**lanning using Py**TOR**ch).

Examples using Common Utility Functions

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 $\mathcal{U}(X; \beta) = -\frac{1}{\beta}\log \mathbb{E}\left[e^{-\beta X}\right]$, where $\beta \in \mathbb{R}$ is a fixed parameter that controls the level of risk aversion (Bäuerle and Rieder 2014). Applying (5):

$$\frac{\partial}{\partial \mathbf{a}_t}\mathcal{U}_{\mathbf{s}_{1:H}\sim p}(Z(\mathbf{a}_{0:H}); \beta) \propto \frac{\partial}{\partial \mathbf{a}_t}\mathbb{E}_{\mathbf{s}_{1:H}\sim p}\left[e^{-\beta Z(\mathbf{a}_{0:H})}\right]$$

$$= \frac{\partial}{\partial \mathbf{a}_t}\mathbb{E}_{\xi_{0:H-1}\sim f_\xi}\left[e^{-\beta Z_\xi(\mathbf{a}_{0:H})}\right]$$

$$= \mathbb{E}_{\xi_{0:H-1}\sim f_\xi}\left[\frac{\partial}{\partial \mathbf{a}_t} e^{-\beta Z_\xi(\mathbf{a}_{0:H})}\right] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{a}_t} e^{-\beta \hat{v}^{(i)}}.$$

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

Risk-Aware Planning with Mean-Variance Approximation The calculation of the entropic utility is susceptible to overflow (Gosavi, Das, and Murray 2014). The mean-variance objective, $\mathcal{U}(X; \beta) \approx \mathbb{E}[X] + \frac{\beta}{2}\mathrm{Var}[X]$, which can be derived directly from the Taylor expansion of the entropic utility, offers a numerically stable alternative. RAPTOR can be easily applied by backpropagating through the moments of the return:

$$\frac{\partial}{\partial \mathbf{a}_t}\mathbb{E}_{\mathbf{s}_{1:H}\sim p}[Z(\mathbf{a}_{0:H})] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{a}_t}\hat{v}^{(i)}$$

$$\frac{\partial}{\partial \mathbf{a}_t}\mathrm{Var}_{\mathbf{s}_{1:H}\sim p}[Z(\mathbf{a}_{0:H})] \approx \frac{1}{N-1}\sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{a}_t}\left(\hat{v}^{(i)} - \frac{1}{N}\sum_{j=1}^{N}\hat{v}^{(j)}\right)^2.$$

Thus, g has an obvious representation in terms of the sample mean and variance of the return distribution, and $w_i \propto 1 + \beta(\hat{v}^{(i)} - \frac{1}{N}\sum_{j=1}^{N}\hat{v}^{(j)})$ assigns more weight to returns that lie further away from the empirical mean.

Risk-Aware Planning with CVaR CVaR is defined as the expectation over the lower α-percentile of the return distribution, $\mathrm{CVaR}_\alpha[X] = \mathbb{E}[X | X \leq \mathrm{VaR}_\alpha[X]]$ where $\mathrm{VaR}_\alpha[X]$ is the α-percentile of the distribution of X (Chow et al. 2015).

Unfortunately, $\mathrm{VaR}_\alpha[X]$ is not differentiable, so we apply the *straight-through gradient* trick (Bengio, Léonard, and Courville 2013; Courbariaux et al. 2016). Specifically, RAPTOR sorts $\hat{v}^{(1)}, \dots \hat{v}^{(N)}$ to obtain order statistics $\hat{v}^{(i_1)} \leq \hat{v}^{(i_2)} \leq \dots \leq \hat{v}^{(i_N)}$, uses them to estimate VaR_α as $\hat{v}^{(i_{\lfloor N(1-\alpha) \rfloor})}$, and then backpropagates through the smallest $\lfloor N(1-\alpha) \rfloor$ samples without passing the gradient through $\hat{v}^{(i_{\lfloor N(1-\alpha) \rfloor})}$ (Kolla et al. 2019):

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{a}_t} \mathrm{CVaR}_\alpha[Z(\mathbf{a}_{0:H})] \\
&\approx \frac{1}{N(1-\alpha)} \frac{\partial}{\partial \mathbf{a}_t} \sum_{i=1}^{N} \hat{v}^{(i)} \mathbb{1}\left[\hat{v}^{(i)} \leq \hat{v}^{(i_{\lfloor N(1-\alpha) \rfloor})}\right] \\
&\approx \frac{1}{N(1-\alpha)} \sum_{j=1}^{\lfloor N(1-\alpha) \rfloor} \frac{\partial}{\partial \mathbf{a}_t} \hat{v}^{(i_j)}.
\end{aligned}$$

This corresponds to $g : (\hat{v}^{(1)}, \dots \hat{v}^{(N)}) \mapsto \frac{1}{N(1-\alpha)} \sum_{i=1}^{N} \hat{v}^{(i)} \mathbb{1}\left[\hat{v}^{(i)} \leq \hat{v}^{(i_{\lfloor N(1-\alpha) \rfloor})}\right]$ with the approximated gradient above. We also have $w_i = 0$ for those samples $\hat{v}^{(i)}$ that lie above VaR_α, and $w_i \propto 1$ 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 $\hat{v}^{(i)}$, such as quadratic utility (Luenberger et al. 1997), (mean-)semideviation/semivariance (Ogryczak and Ruszczyński 1999), and optimized certainty equivalents (Ben-Tal and Teboulle 2007) have also been studied in the literature, and could be handled quite easily within our framework.

Risk-Aware Planning for Closed-Loop Policies

The optimization of $U(\mathbf{a}_{0:H})$ provides a maximizing plan $\mathbf{a}^*_{0:H}$, that commits to a fixed sequence of actions starting from $\mathbf{s}_0$ 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 $U(\mathbf{a}^*_{0:H})$ 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; Issakkimuthu 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) $\pi_\theta : \mathcal{S} \to \mathcal{A}$ (Bueno et al. 2019) by maximizing

$$U(\theta) = \mathcal{U}_{\mathbf{s}_{1:H} \sim p}\left(\sum_{t=0}^{H} r(\mathbf{s}_t, \pi_\theta(\mathbf{s}_t))\right)$$

over θ. 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 $Z(\theta) := \sum_{t=0}^{H} r(\mathbf{s}_t, \pi_\theta(\mathbf{s}_t))$ has a complex dependence on θ. However, the distributional perspective of SLPs also extends to DRPs. In particular:

$$\begin{aligned}
\mathbf{s}_t &= \phi(\mathbf{s}_{t-1}, \pi_\theta(\mathbf{s}_{t-1}), \xi_{t-1}) \\
&= \phi\big(\phi(\mathbf{s}_{t-2}, \pi_\theta(\mathbf{s}_{t-2}), \xi_{t-1}), \\
&\qquad \pi_\theta\left(\phi(\mathbf{s}_{t-2}, \pi_\theta(\mathbf{s}_{t-2}), \xi_{t-1})\right), \xi_{t-1}\big) \\
&= \dots := \Psi_t(\mathbf{s}_0, \theta, \xi_{0:t-1}),
\end{aligned}$$

and the return distribution can now be written concisely as

$$Z_\xi(\theta) = \sum_{t=0}^{H} r(\Psi_t(\mathbf{s}_0, \theta, \xi_{0:t-1}), \pi_\theta(\Psi_t(\mathbf{s}_0, \theta, \xi_{0:t-1}))).$$

Therefore, the gradient of the sample $\hat{v}^{(i)}(\theta) \sim Z(\theta)$ 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 backpropagating through the approximated utility $\hat{U}(\mathbf{a}_{0:H}) = g(\hat{v}^{(1)}, \dots \hat{v}^{(N)})$, 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 Findeisen 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 $\mathrm{crossing}_t$ denote the length of a sub-trajectory

that crosses the zone when moving from $\mathbf{s}_t$ to $\mathbf{s}_t + \mathbf{a}_t$. Then, a normally distributed noise $\mathcal{N}(0, (\text{crossing}_t \cdot \sigma_h)^2)$ is added to the next state. On the other hand, when $\text{crossing}_t = 0$, a small noise $\mathcal{N}(0, \sigma_l^2)$ with variance $\sigma_l^2 \ll \sigma_h^2$ is added. Thus, the transition function can be reparameterized as:

$$\phi(\mathbf{s}_t, \mathbf{a}_t, \xi_t) = \mathbf{s}_t + \mathbf{a}_t + \text{crossing}_t \sigma_h \xi_t + \mathbb{1}[\text{crossing}_t = 0]\sigma_l \xi_t, \quad \xi_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{2\times 2}).$$

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 $a_{t,ij} = [\mathbf{a}_t]_{ij}$ 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., $\sum_j a_{t,ij} \leq s_{t,i}$. The goal of the agent is to keep the water level within safe limits $[l_i, u_i]$, so we define the reward function as $r(\mathbf{s}_t, \mathbf{a}_t) = \sum_{i=1}^N R_i$, where

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

and where P_u (P_l) is the penalty for water levels above (below) the upper (lower) bound. Note that P_u is weighted more heavily than P_l, 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 ξ_t with rate λ. Thus, switching to a component-wise view of ϕ_i for notational clarity, the transition function for each reservoir i becomes

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

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, $\mathbf{a}_t$, 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 $r(\mathbf{s}_t, \mathbf{a}_t) = \sum_{i=1}^N R_i$, where

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

and where T_i is the target temperature for room i, T_l is the lowest acceptable temperature, and P_u is a penalty applied when the temperature drops below T_l. As a cost-saving measure, T_i is set near T_l 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 $\mathcal{N}(0, \sigma_{ij}^2)$. Combining this, the reparameterized dynamics for room i are

$$\phi_i(\mathbf{s}_t, \mathbf{a}_t, \xi_t) = s_{t,i} + a_{t,i}(T_c - s_{t,i}) + \sigma_a \xi_t + \sum_{j \in A(i)} \frac{(s_{t,j} - s_{t,i})^3}{\rho_{ij}} + \sigma_{ij}\xi_t + T_o + \sigma_o \xi_t,$$

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

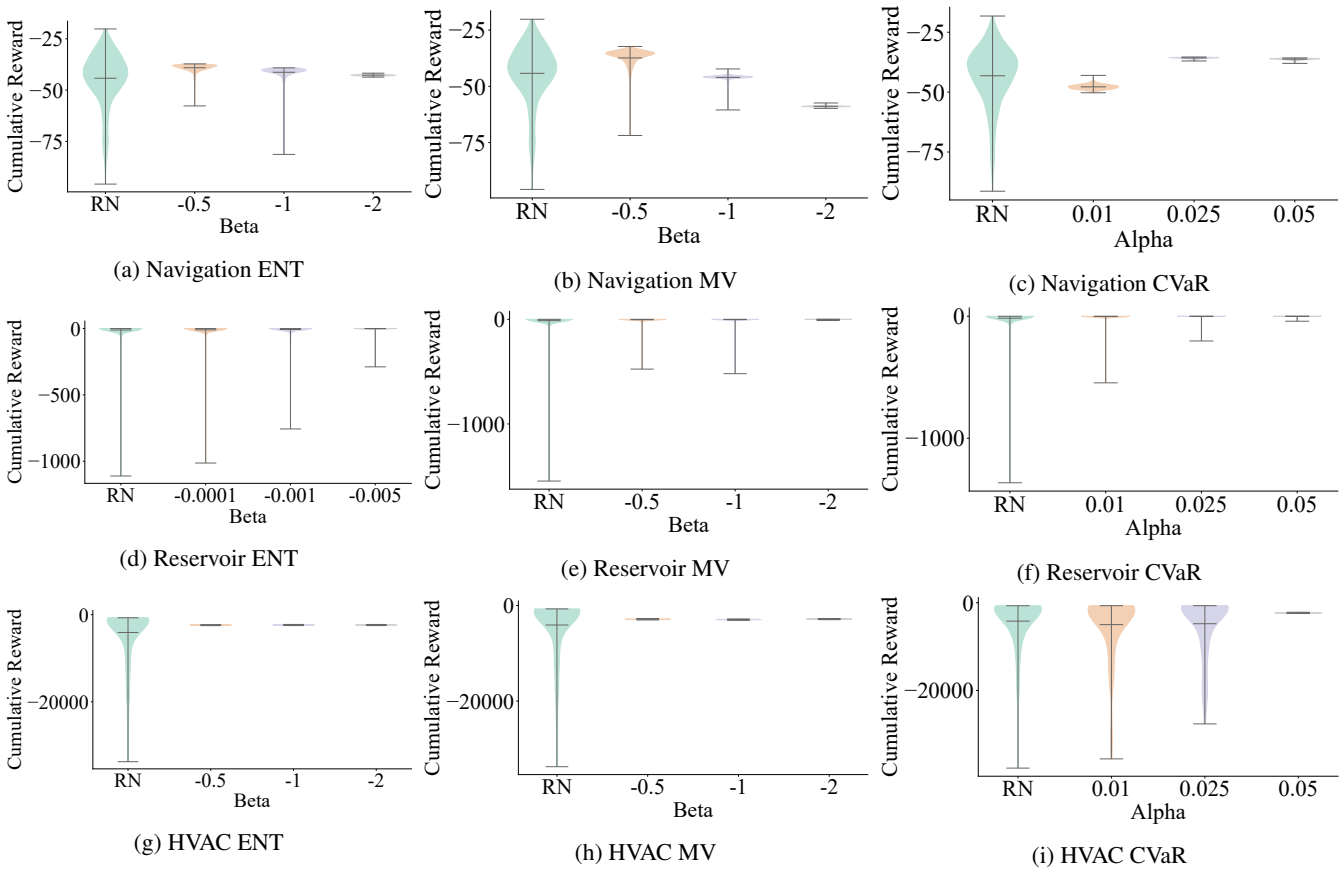
Performance of Risk-Aware Planning

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 (β for MV and ENT, α for CVaR) to see the effects of increasing risk-aversion. Indeed, as the magnitude of β and α 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, which CVaR 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, RAPTOR-SLP with CVaR is particularly adept at controlling the tail distribution and the worst-case outcomes, as can



(a) Navigation ENT (b) Navigation MV (c) Navigation CVaR

(d) Reservoir ENT (e) Reservoir MV (f) Reservoir CVaR

(g) HVAC ENT (h) HVAC MV (i) HVAC CVaR

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 α is small, potentially because low α discards many samples (recall that CVaR sets $w_i = 0$ for samples $\hat{v}^{(i)}$ above VaR_α).

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 workaround 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.

References

Bäuerle, N.; and Rieder, U. 2014. More risk-sensitive Markov decision processes. *Mathematics of Operations Research*, 39(1): 105–120.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. In *ICML*, 449–458. PMLR.

Ben-Tal, A.; and Teboulle, M. 2007. An old-new concept of convex risk measures: The optimized certainty equivalent. *Mathematical Finance*, 17(3): 449–476.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

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.

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.

Burns, E.; Benton, J.; Ruml, W.; Yoon, S.; and Do, M. 2012. Anticipatory on-line planning. In *ICAPS*, volume 22.

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

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Defourny, B.; Ernst, D.; and Wehenkel, L. 2008. Risk-aware decision making and dynamic programming. In *NeurIPS 2008 Workshop on Model Uncertainty and Risk in RL*.

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.

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

Fishburn, P. C. 1970. Utility theory for decision making. Technical report, Research analysis corp McLean VA.

Gosavi, A. A.; Das, S. K.; and Murray, S. L. 2014. Beyond exponential utility functions: A variance-adjusted approach for risk-averse reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 1–8. IEEE.

Issakkimuthu, M.; Fern, A.; Khardon, R.; Tadepalli, P.; and Xue, S. 2015. Hindsight optimization for probabilistic planning with factored actions. In *ICAPS*, volume 25.

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

Kolla, R. K.; Prashanth, L.; Bhat, S. P.; and Jagannathan, K. 2019. Concentration bounds for empirical conditional value-at-risk: The unbounded case. *Operations Research Letters*, 47(1): 16–20.

Luenberger, D. G.; et al. 1997. Investment science. *OUP Catalogue*.

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

Mercier, L.; and Van Hentenryck, P. 2008. Amsaa: A multi-step anticipatory algorithm for online stochastic combinatorial optimization. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, 173–187. Springer.

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

Ogryczak, W.; and Ruszczyński, A. 1999. From stochastic dominance to mean-risk models: Semideviations as risk measures. *European journal of operational research*, 116(1): 33–50.

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.

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

Ruiz, F.; Titsias, M.; and Blei, D. 2016. The generalized reparameterization gradient. *NeurIPS*, 460–468.

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.

Stigler, G. J. 1950. The development of utility theory. I. *Journal of political economy*, 58(4): 307–327.

Tamar, A.; Di Castro, D.; and Mannor, S. 2012. Policy gradients with variance related risk criteria. In *ICML*, 1651–1658.

Tamar, A.; Di Castro, D.; and Mannor, S. 2016. Learning the variance of the reward-to-go. *The Journal of Machine Learning Research*, 17(1): 361–396.

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.

Appendix

A Proof of Theorem 1

It is easy to check that the entropic utility $\mathcal{U}_\xi(\cdot;\beta) = -\frac{1}{\beta}\log \mathbb{E}_\xi\left[e^{-\beta X}\right]$ satisfies the properties of monotonicity, translation invariance, and concavity, e.g. it is a concave utility function (Ruszczyński 2010). Next, let ξ be a random variable, A be an arbitrary set and $X_a = g_a(\xi)$ for $a \in A$ and some functions g_a. Since $\mathbb{P}(\sup_{a\in A} X_a \geq X_b) = 1$ for every $b \in A$ and $\mathcal{U}_\xi(\cdot;\beta)$ is monotone, it follows that $\mathcal{U}_\xi(\sup_{a\in A} X_a;\beta) \geq \mathcal{U}_\xi(X_b;\beta)$. Since b is arbitrary, we have $\mathcal{U}_\xi(\sup_{a\in A} X_a;\beta) \geq \sup_{a\in A}\mathcal{U}_\xi(X_a;\beta)$.

Next, observe that the utility of the optimal closed-loop Markov policy satisfies (Bäuerle and Rieder 2014)

$$U_t^*(\mathbf{s}_t) = \sup_{\mathbf{a}} \mathcal{U}_{\mathbf{s}_{t+1}\sim p}\big(r(\mathbf{s}_t,\mathbf{a}) + U_{t+1}^*(\mathbf{s}_{t+1})\big)\,.$$

Using the monotonicity and translation invariance properties, the recursiveness of the entropic utility (Osogami 2012), and the reparameterization $\mathbf{s}_{t+1} = \phi(\mathbf{s}_t, \mathbf{a}_t, \xi_t)$, we have:

$$\begin{aligned}
U_0^*(\mathbf{s}_0) &= \sup_{\mathbf{a}_0} \mathcal{U}_{\xi_0}(r(\mathbf{s}_0,\mathbf{a}_0) + U_1^*(\mathbf{s}_1)) \\
&= \sup_{\mathbf{a}_0} \mathcal{U}_{\xi_0}\left(r(\mathbf{s}_0,\mathbf{a}_0) + \sup_{\mathbf{a}_1} \mathcal{U}_{\xi_1}(r(\mathbf{s}_1,\mathbf{a}_1) + U_2^*(\mathbf{s}_2))\right) \\
&\geq \sup_{\mathbf{a}_0}\sup_{\mathbf{a}_1} \mathcal{U}_{\xi_0}(r(\mathbf{s}_0,\mathbf{a}_0) + \mathcal{U}_{\xi_1}(r(\mathbf{s}_1,\mathbf{a}_1) + U_2^*(\mathbf{s}_2))) \\
&= \sup_{\mathbf{a}_0,\mathbf{a}_1} \mathcal{U}_{\xi_0}(\mathcal{U}_{\xi_1}(r(\mathbf{s}_0,\mathbf{a}_0) + r(\mathbf{s}_1,\mathbf{a}_1) + U_2^*(\mathbf{s}_2))) \\
&= \sup_{\mathbf{a}_0,\mathbf{a}_1} \mathcal{U}_{\xi_0,\xi_1}(r(\mathbf{s}_0,\mathbf{a}_0) + r(\mathbf{s}_1,\mathbf{a}_1) + U_2^*(\mathbf{s}_2)) \\
&\geq \dots \\
&= \sup_{\mathbf{a}_{0:H}} \mathcal{U}_{\xi_{0:H}}(r(\mathbf{s}_0,\mathbf{a}_0) + \dots + r(\mathbf{s}_H,\mathbf{a}_H)) \\
&= \sup_{\mathbf{a}_{0:H}} \mathcal{U}_{\xi_{0:H}}(Z(\mathbf{a}_{0:H})) \\
&= U(\mathbf{a}_{0:H}^*),
\end{aligned}$$

where the last line is the utility of an optimal open-loop policy. This completes the proof. □

B Additional Experimental Results

In this part, we provide and discuss some additional experimental results on the three domains we introduced in the main text. Firstly, we present the analysis for DRP, comparing risk-neutral (RN) agents against RAPTOR-DRP with different risk utilities (ENT, MV, and CVaR). Then, we plot progressions of states in Reservoir and HVAC domains, which have been obtained by following the plans that RN or RAPTOR produced.

DRP Results Figure 4 shows the return distributions obtained by DRP agents with four different utilities (RN, ENT, MV, and CVaR) on Navigation (Figure 4a-4c), Reservoir (Figure 4d-4f), and HVAC (Figure 4g-4i) while varying the risk-aversion parameters (α, β) as in Figure 3.

In all three domains, RN-DRP performs substantially better than RN-SLP. That is, RN-DRP achieves higher expected returns while having low variances in the returns. The boost in performance is expected because in principle, RN-DRP is able to react to a stochastic environment in a closed-loop manner, whereas RN-SLP sticks to a fixed plan that does not consider actual state realizations over time (Bueno et al. 2019).

Despite the improvements given by RN-DRP in domains with high-stakes decisions — where large penalties can be incurred if not careful (i.e., Reservoir and HVAC), we clearly see that RN-DRP achieves lower reward distribution than all three variants (ENT, MV, CVaR) of RAPTOR-DRP. Here, the pattern of reduced variances with increased magnitude of risk-aversion parameters is not as much clear as in RAPTOR-SLP. However, the results still imply that we can find appropriate risk-aversion parameters such that we can reduce the variability in returns and prevent worst-case outcomes without suffering much (or even increasing) in expected returns. Also, as discussed previously, it is often harder to optimize RAPTOR-DRP than RAPTOR-SLP since the reactive policy network introduces additional training parameters and must learn to generalize across the entire state space. In other words, better network optimization may lead to improved RAPTOR-DRP results. Below, we provide detailed breakdown of the results.

In **Navigation**, we can see that increasing the magnitude of α or β parameters generally results in reduced variances in the returns. However, the pattern is not as clear compared to Figure 3g-3i for RAPTOR-SLP (note: the ranges of return values are different). When using the CVaR utility, we found optimizing RAPTOR-DRP somewhat challenging. On the other hand, ENT shows a better trade-off with respect to the β parameter. When considering the actual trajectories traversed by RAPTOR-DRP agents, we observe that none of them failed to reach the goal region, showing the effectiveness of the optimized reactive policies.

In **Reservoir**, we observe a significantly better worst-case outcome of RN-DRP agent compared to the SLP result (from roughly -1000 in RN-SLP to -150 in RN-DRP). The improvement is much more pronounced in RAPTOR-DRP agents: the range is now between about -4 and 0 with MV and minimal with CVaR objective (note: ENT with $\beta = -0.001$ showed comparably large negative worst-case outcome, but the results are generally reliable with other β values while still substantially better than RAPTOR-SLP with ENT). For all α values in all sampled scenarios, RAPTOR-DRP with CVaR is able to avoid the large cost due to overflows, showing the effectiveness of RAPTOR.

From the results on **HVAC**, we observe that (1) the return distributions obtained by different RAPTOR-DRP agents have similar variances, (2) RN-DRP has considerably improved return values compared to its SLP counterpart, (3) RAPTOR-DRP agents have also generally improved the expected returns compared to RAPTOR-SLP, (4) increasing the risk-aversion parameter does not necessarily produce the same impact on different risk utilities, and (5) RAPTOR-DRP always achieves higher returns than RN-DRP.

Example State Trajectories We previously provided example state trajectories under different risk metrics for SLP and DRP for Navigation in Figure 1. In this section, we do the same for Reservoir and HVAC.

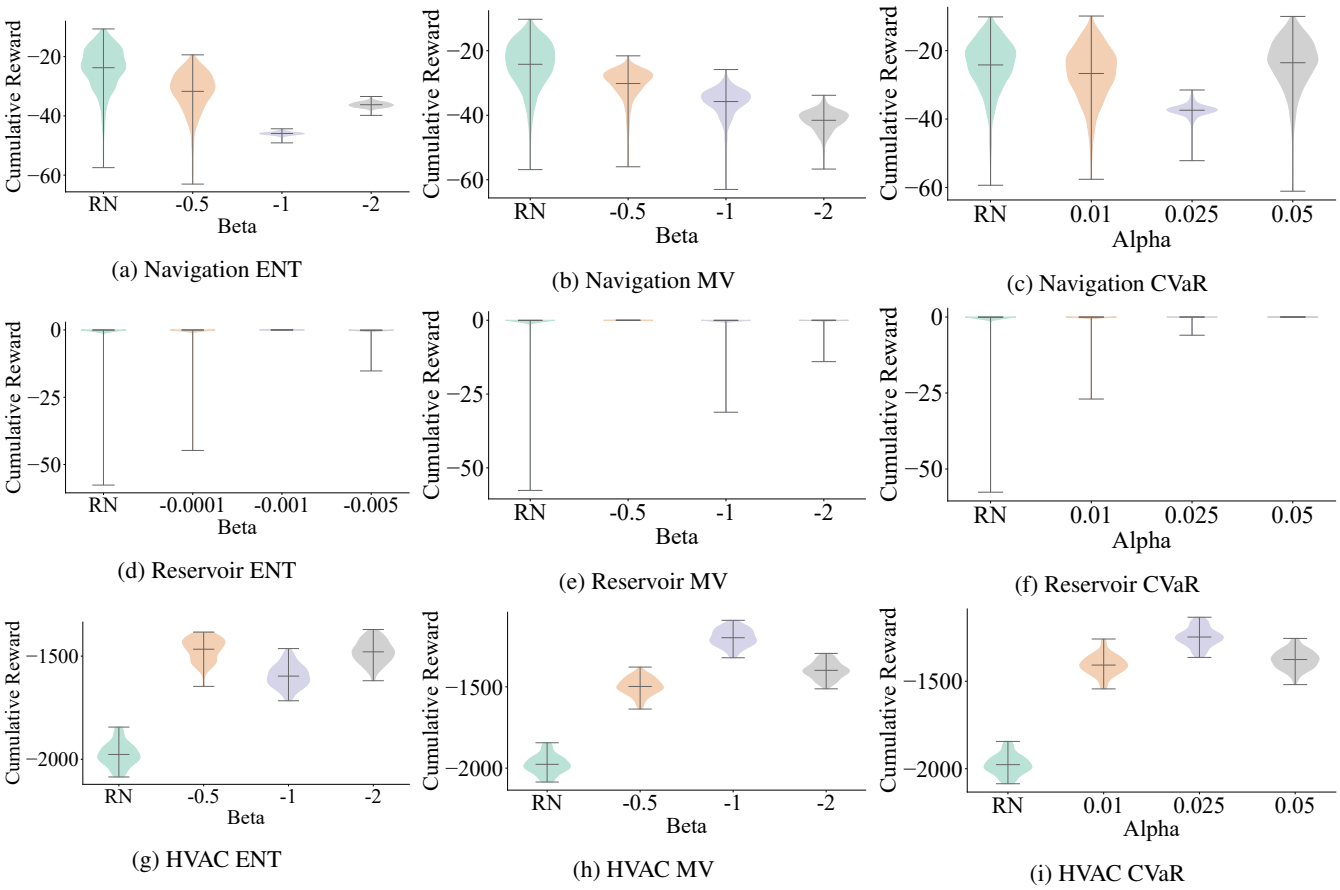
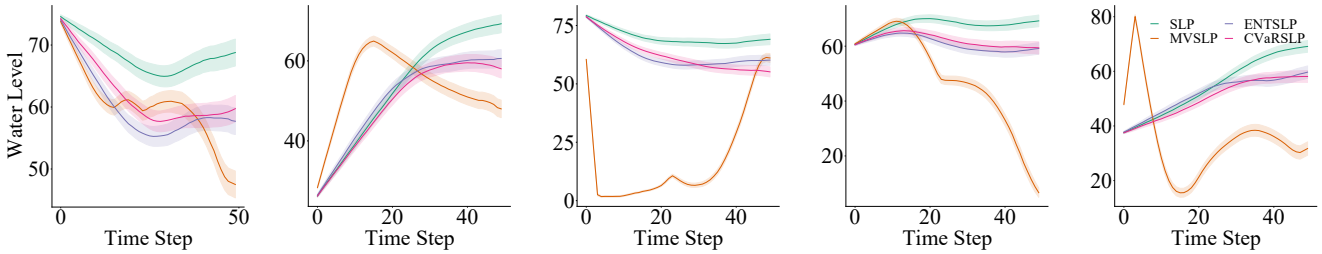


Figure 4: The cumulative reward distributions obtained by risk-neutral DRP (RN) and RAPTOR-DRP agents with three different risk utilities (ENT, MV, and CVaR) in Navigation (4a-4c), Reservoir (4d-4f), and HVAC (4g-4i) domains.

Figure 5 and Figure 6 show the evolution of water levels in different reservoirs controlled by SLP and DRP agents, respectively. We observe that RN-SLP almost always maintains higher water levels than RAPTOR-SLP in all reservoirs, which has led to overflows and large penalties, resulting in poor worst-case outcomes as we saw in Figure 3. On the other hand, the optimized reactive policies shown in Figure 6 aren't as easy as the SLP results to interpret. In general, when the initial water level of a reservoir is close to the upper limit, DRPs tend to bring it down; while when it starts low, it either remains at about the same level or gets increased. For Reservoir 1, 3, and 4 where the initial water levels are comparably high, RN-DRP does not let the water flow downstream fast enough, which might have occasionally caused overflows. This is reflected in the return distribution of RN-DRP, which has a longer tail than RAPTOR-DRP distributions.

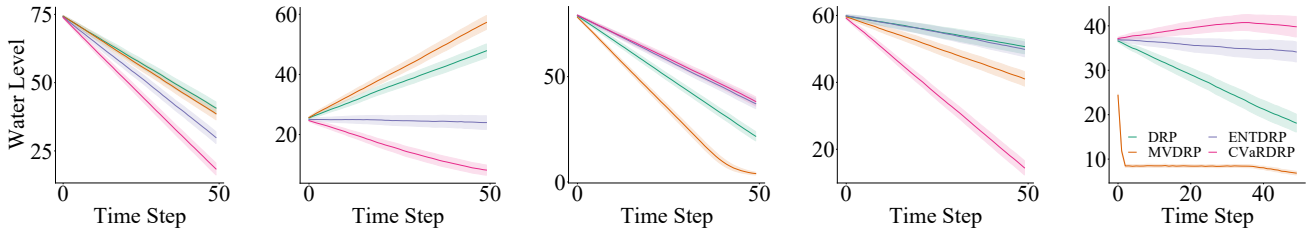
Figure 7 and Figure 8 compare how the temperatures in different rooms evolve over time when controlled by different SLP and DRP agents, respectively. As RN-SLP does not take risk into account, it ignores the possibility of occasional large temperature drops while trying to reduce heating costs. Hence, the agent keeps the temperature in all rooms lower than what RAPTOR-SLP agents do. However, this can incur the large negative reward for temperatures going below P_l as exhibited by Figure 3h. The optimized straight line plans for ENT and CVaR turn out to be almost the same, while MV retains slightly higher temperatures throughout all rooms.

The optimized reactive policy for RN-DRP does the opposite with respect to RN-SLP when we look at Figure 8. That is, the agent tries to keep the temperatures high in all rooms. The return distribution in Figure 4h shows that this enabled RN-DRP to avoid the large negative rewards. However, RN-DRP obtains smaller expected returns than RAPTOR-DRP due to increased heating cost and the penalty for deviating from the set temperature. However, the fact that the risk-neutral agent tries to maintain the temperature even higher than what risk-aware agents do is somewhat counter-intuitive. As discussed earlier, this may be related to the difficulty in non-convex optimization of DRPs, which have to generalize across the entire state space. During the optimization of the RN-DRP parameters, we early stop training before the loss jumps up. Otherwise, we found that the resulting policy performed too badly. Interestingly, RAPTOR-DRP optimization was not as challenging, and the optimized policies successfully manage risks while obtain higher expected returns at the same time. Furthermore, we observe that DRP agents are able to keep the temperatures stable over time, while temperatures fluctuate much more when we look at the SLP results.



(a) Reservoir 1 (b) Reservoir 2 (c) Reservoir 3 (d) Reservoir 4 (e) Reservoir 5

Figure 5: The mean water levels and associated noise is shown for each of the five reservoirs with SLP, Mean-Variance SLP ($\beta = -2$), Entropic SLP ($\beta = -0.005$) and CVaR SLP ($\alpha = 0.05$).



(a) Reservoir 1 (b) Reservoir 2 (c) Reservoir 3 (d) Reservoir 4 (e) Reservoir 5

Figure 6: The mean water levels and associated noise is shown for each of the five reservoirs with DRP, Mean-Variance DRP ($\beta = -2$), Entropic DRP ($\beta = -0.0001$) and CVaR DRP ($\alpha = 0.025$).

C Detail of Experiments

Experimental Setting Details In this section, we provide a detailed accounting of the experimental parameters used in all experiments. All experiments were run on a single compute node with AMD Ryzen 7 4800H processor and NVIDIA GeForce RTX 2060 graphics card using PyTorch v1.7.1 and Python version 3.8.5. On this machine, compute time for each training run of a domain took less than 3 minutes.

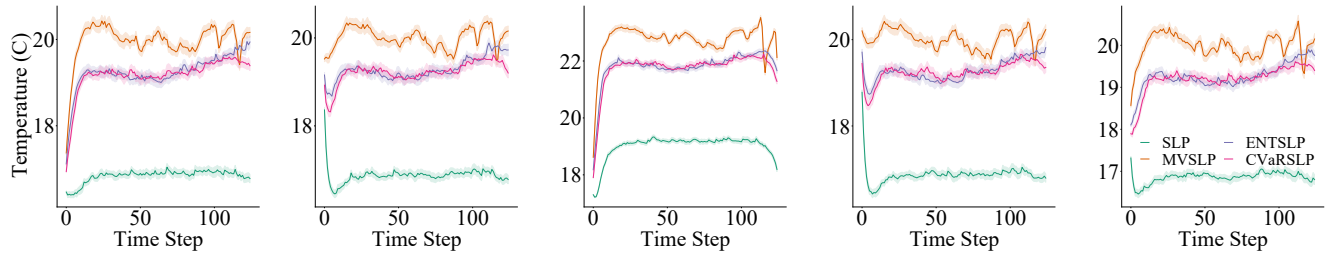
For hyperparameter tuning, β, α, and the learning rates were chosen (as shown in Table 1) by performing grid search of β crossed with learning rate for mean-variance and entropic risk measures, and α crossed with learning rate for CVaR. The learning rate was the main focus of hyperparameter tuning for each risk parameter as it had the greatest sensitivity when compared to other hyperparameters. The results of the grid search were evaluated on expected reward and the values used for grid search are shown in Table 2. Each domain was trained over 201 training epochs, which was sufficient for convergence of all agents. All of the DRP agents had hidden-layer-nodes of $256, 128, 64, 32$. The final layer of this network uses linear activation and all other layers use ReLU activation. This architecture was chosen from the risk-neutral architecture in Bueno et al. (AAAI 2019). The Navigation domain has a time horizon of 20 steps, and agents were trained on a batch size of 4096. For Reservoir, agents were trained on a batch size of 1024 over a 50 time step horizon. Lastly, for HVAC, agents again were trained on a batch size of 4096, but with a time horizon 125 steps.

Domain	Risk-Measure	α/β (Learning Rate)
Navigation	RN SLP	0 (0.5)
	MV SLP	-0.5 (0.05), -1 (0.05), -2 (0.005)
	ENT SLP	-0.5 (0.1), -1 (0.5), -2 (0.5)
	CVaR SLP	0.01 (0.005), 0.025 (0.05), 0.05 (0.1)
	RN DRP	0 (0.005)
	MV DRP	-0.5 (0.01), -1 (0.01), -2 (0.05)
	ENT DRP	-0.5 (0.0001), -1 (0.0001), -2 (0.0005)
	CVaR DRP	0.01 (0.0001), 0.025 (0.0001), 0.05 (0.001)
Reservoir	RN SLP	0 (0.1)
	MV SLP	-0.5 (0.1), -1 (0.1), -2 (0.5)
	ENT SLP	-0.0001 (0.1), -0.001 (0.1), -0.005 (0.1)
	CVaR SLP	0.01 (0.005), 0.025 (0.01), 0.05 (0.005)
	RN DRP	0 (0.005)
	MV DRP	-0.5 (0.005), -1 (0.01), -2 (0.01)
	ENT DRP	-0.0001 (0.005), -0.001 (0.01), -0.005 (0.05)
	CVaR DRP	0.01 (0.005), 0.025 (0.01), 0.05 (0.005)
HVAC	RN SLP	0 (0.01)
	MV SLP	-0.5 (0.1), -1 (0.1), -2 (0.1)
	ENT SLP	-0.5 (0.1), -1 (0.1), -2 (0.1)
	CVaR SLP	0.01 (0.01), 0.025 (0.01), 0.05 (0.1)
	RN DRP	0 (0.05)
	MV DRP	-0.5 (0.005), -1 (0.01), -2 (0.005)
	ENT DRP	-0.5 (0.01), -1 (0.01), -2 (0.0005)
	CVaR DRP	0.01 (0.001), 0.025 (0.0005), 0.05 (0.0005)

Table 1: The hyperparameters chosen for the reward distribution figures are shown for each domain and each type of risk measure. For RN, MV, and ENT β is shown in the α/β column, and for CVaR α is shown in this column. The corresponding learning rate is shown in parentheses for each risk parameter.

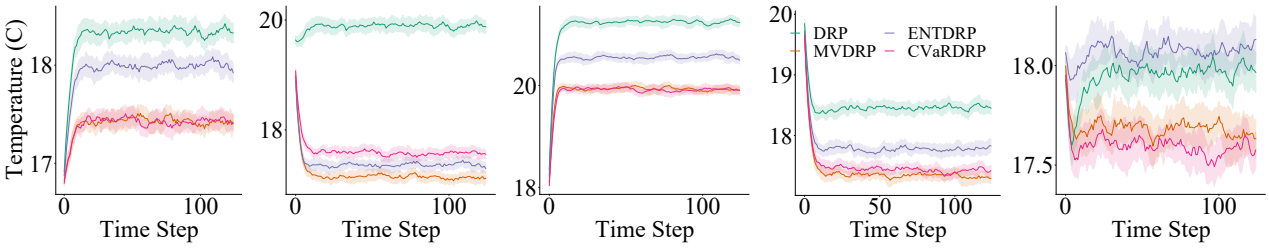
Domain	Risk-Measure	α/β	Learning Rate
Navigation	RN	0	$0.0001, 0.0005, \ldots, 0.1, 0.5$
	MV	$-0.1, -0.5, -1, -2$	$0.0001, 0.0005, \ldots, 0.1, 0.5$
	ENT	$-0.1, -0.5, -1, -2$	$0.0001, 0.0005, \ldots, 0.1, 0.5$
	CVaR	$0.01, 0.025, 0.05$	$0.0001, 0.0005, \ldots, 0.1, 0.5$
Reservoir	RN	0	$0.0001, 0.0005, \ldots, 0.05, 0.1$
	MV	$-0.1, -0.5, -1, -2$	$0.0001, 0.0005, \ldots, 0.05, 0.1$
	ENT	$-0.0001, -0.0005, -0.001, -0.005$	$0.0001, 0.0005, \ldots, 0.05, 0.1$
	CVaR	$0.01, 0.025, 0.05$	$0.0001, 0.0005, \ldots, 0.05, 0.1$
HVAC	RN	0	$0.0001, 0.0005, \ldots, 0.05, 0.1$
	MV	$-0.1, -0.5, -1, -2$	$0.0001, 0.0005, \ldots, 0.05, 0.1$
	ENT	$-0.1, -0.5, -1, -2$	$0.0001, 0.0005, \ldots, 0.05, 0.1$
	CVaR	$0.01, 0.025, 0.05$	$0.0001, 0.0005, \ldots, 0.05, 0.1$

Table 2: The hyperparameter grid search values are shown for each domain and each type of risk measure. For RN, MV, and ENT β is shown in the α/β column, and for CVaR α is shown in this column.



(a) Room 1 (b) Room 2 (c) Room 3 (d) Room 4 (e) Room 5

Figure 7: The mean room temperatures and associated noise is shown for each of the five rooms with SLP, Mean-Variance SLP ($\beta = -2$), Entropic SLP ($\beta = -2$) and CVaR SLP ($\alpha = 0.05$).



(a) Room 1 (b) Room 2 (c) Room 3 (d) Room 4 (e) Room 5

Figure 8: The mean room temperatures and associated noise is shown for each of the five rooms with DRP, Mean-Variance DRP ($\beta = -1$), Entropic DRP ($\beta = -0.5$) and CVaR DRP ($\alpha = 0.025$).