2023

Reliable deep reinforcement learning: stable training and robust deployment

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

RELIABLE DEEP REINFORCEMENT LEARNING: STABLE TRAINING AND ROBUST DEPLOYMENT

by

JAMES QUEENEY

B.A., Colgate University, 2013
M.S., Boston University, 2022

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2023

Approved by

First Reader

Ioannis Ch. Paschalidis, PhD
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering
Professor of Systems Engineering
Professor of Computing & Data Sciences

Second Reader

Christos G. Cassandras, PhD
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor and Division Head of Systems Engineering

Third Reader

Ashok Cutkosky, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Systems Engineering
Assistant Professor of Computer Science

Fourth Reader

Eshed Ohn-Bar, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Computer Science

Fifth Reader

Mouhacine Benosman, PhD
Senior Principal Research Scientist
Mitsubishi Electric Research Laboratories

Acknowledgments

I am grateful for all of the support I have received throughout the course of my PhD. First and foremost, I would like to thank my advisors Yannis Paschalidis and Christos Cassandras. They allowed me the freedom to pursue my research interests, while providing incredible guidance and mentorship at every step of the way. I would also like to recognize Ashok Cutkosky, Eshed Ohn-Bar, and Mouhacine Benosman for their time and feedback as members of my dissertation committee, and Alex Olshevsky for serving as committee chair. In addition, I had the privilege of collaborating with Mouhacine during my PhD, and I have benefited greatly from his insights and mentorship. I am grateful to all of my other collaborators, my PhD cohort, and my fellow lab members for their invaluable help and thoughtful discussions over the past five years. I also appreciate the staff of the Division of Systems Engineering and the Center for Information & Systems Engineering for everything they have done for me throughout my time at Boston University. Finally, I am thankful for the constant encouragement from my family and friends. Most importantly, I would like to thank my parents Jim and Pat Queeney and my wife Quincey Spagnoletti for their unconditional love and unwavering support. I am truly lucky to have such amazing people in my life.

RELIABLE DEEP REINFORCEMENT LEARNING: STABLE TRAINING AND ROBUST DEPLOYMENT

JAMES QUEENEY

Boston University, College of Engineering, 2023

Major Professors: Ioannis Ch. Paschalidis, PhD
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering
Professor of Systems Engineering
Professor of Computing & Data Sciences

Christos G. Cassandras, PhD
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor and Division Head of Systems Engineering

ABSTRACT

Deep reinforcement learning (RL) represents a data-driven framework for sequential decision making that has demonstrated the ability to solve challenging control tasks. This data-driven, learning-based approach offers the potential to improve operations in complex systems, but only if it can be trusted to produce reliable performance both during training and upon deployment. These requirements have hindered the adoption of deep RL in many real-world applications. In order to overcome the limitations of existing methods, this dissertation introduces reliable deep RL algorithms that deliver (i) stable training from limited data and (ii) robust, safe deployment in the presence of uncertainty.

The first part of the dissertation addresses the interactive nature of deep RL,

where learning requires data collection from the environment. This interactive process can be expensive, time-consuming, and dangerous in many real-world settings, which motivates the need for reliable and efficient learning. We develop deep RL algorithms that guarantee stable performance throughout training, while also directly considering data efficiency in their design. These algorithms are supported by novel policy improvement lower bounds that account for finite-sample estimation error and sample reuse.

The second part of the dissertation focuses on the uncertainty present in real-world applications, which can impact the performance and safety of learned control policies. In order to reliably deploy deep RL in the presence of uncertainty, we introduce frameworks that incorporate safety constraints and provide robustness to general disturbances in the environment. Importantly, these frameworks make limited assumptions on the training process, and can be implemented in settings that require real-world interaction for training. This motivates deep RL algorithms that deliver robust, safe performance at deployment time, while only using standard data collection from a single training environment.

Overall, this dissertation contributes new techniques to overcome key limitations of deep RL for real-world decision making and control. Experiments across a variety of continuous control tasks demonstrate the effectiveness of our algorithms.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

C-MDP	.....	Constrained MDP
CRPO	.....	Constraint-Rectified Policy Optimization
CVaR	.....	Conditional Value-at-Risk
GAE	.....	Generalized Advantage Estimation
GePPO	.....	Generalized PPO
GeTRPO	.....	Generalized TRPO
GeVMPO	.....	Generalized VMPO
GPI	.....	Generalized Policy Improvement
KL	.....	Kullback-Leibler
KWIK	.....	Knows What It Knows
MDP	.....	Markov Decision Process
MPO	.....	Maximum a Posteriori Policy Optimization
OOD	.....	Out-of-Distribution
OTP	.....	Optimal Transport Perturbations
PPO	.....	Proximal Policy Optimization
RAMU	.....	Risk-Averse Model Uncertainty
RC-MDP	.....	Robust Constrained MDP
RL	.....	Reinforcement Learning
TRPO	.....	Trust Region Policy Optimization
TV	.....	Total Variation
UA-TRPO	.....	Uncertainty-Aware TRPO
VMPO	.....	On-Policy MPO

Chapter 1

Introduction

Reinforcement learning (RL) represents a general framework for data-driven control in sequential decision making settings, where an agent learns how to act by leveraging information from its past experiences. In recent years, deep RL methods have combined the RL framework with deep neural network function approximators to successfully solve a variety of games (Mnih et al., 2015; Schrittwieser et al., 2020) and simulated control tasks (Duan et al., 2016). These impressive results demonstrate the potential of deep RL, and provide hope that these techniques can be used to improve real-world decision making and control.

The data-driven, learning-based approach of deep RL provides a natural pipeline for improving operations within many important application areas, including robotics, autonomous navigation, healthcare, and cyber-physical systems. By using observed data as feedback to iteratively improve decision making and control in these complex settings, deep RL has the potential to significantly benefit society. However, in order to successfully apply deep RL in real-world applications without the need for ongoing human supervision, practitioners must be able to trust that deep RL algorithms will deliver reliable performance both during training and upon deployment. Existing algorithms typically lack such performance guarantees, which has hindered the adoption of deep RL and limited its societal impact.

In order to enable the future use of deep RL in application areas of societal importance, this dissertation contributes new techniques to overcome key limitations of

existing deep RL algorithms. In particular, we introduce *reliable deep RL algorithms* that deliver (i) stable training from limited data and (ii) robust, safe deployment in the presence of uncertainty.

1.1 Stable Training from Limited Data

The interactive nature of deep RL requires data collection to guide learning. In settings where we must deploy the current control policy to collect real-world data during training, poor performance at any point can be both costly and dangerous. In addition, data collection is often expensive and time-consuming in physical real-world settings. As a result, reliable deep RL algorithms should deliver practical performance guarantees throughout training (i.e., *stable training*), while also making efficient use of limited data. The combination of these requirements is not an easy task, as training stability and data efficiency often represent competing interests.

Given the importance of training stability in real-world applications, policy improvement methods represent promising starting points in the design of reliable deep RL algorithms. These methods are motivated by lower bounds on the expected performance loss at every update, providing theoretical support for stable training. Unfortunately, the performance guarantees of existing policy improvement methods require the use of data from the current policy (i.e., on-policy) and only hold in expectation. Therefore, in order for these guarantees to hold in practice, a large number of samples must be collected under the current policy prior to each update. This results in high sample complexity and slow learning, which limits the effectiveness of these algorithms when data collection is difficult.

In Chapter 3 and Chapter 4 of this dissertation, we develop techniques to extend the stable training benefits of policy improvement methods to the limited data setting. We construct novel policy improvement lower bounds that control the error introduced

by (i) sample-based uncertainty and (ii) sample reuse, which motivate reliable deep RL algorithms with stable training and improved data efficiency.

1.2 Robust and Safe Deployment

In addition to stable training, robustness and safety are critical for the reliable deployment of deep RL in real-world decision making applications (Xu et al., 2022). Many physical real-world settings require safe operations, where task completion cannot come at the expense of safety. Moreover, performance and safety are often sensitive to changes in the environment, which are common in real-world deployment scenarios due to unknown disturbances or irreducible modeling errors. Therefore, reliable deep RL algorithms should deliver safe decision making and control even in the presence of uncertainty.

The need for safe operations is typically addressed by incorporating safety constraints into the deep RL framework. However, popular safe RL algorithms only focus on satisfying safety requirements in the training environment. They do not consider irreducible uncertainty about the true environment at deployment time, which we refer to as *model uncertainty*. In order to reliably deploy learned control policies, it is important to account for the impact of model uncertainty on both performance and safety.

Unfortunately, common implementations of model uncertainty in deep RL are not always suitable for real-world decision making settings. These methods often assume access to a detailed simulator during training, and consider very structured forms of model uncertainty based on modifying important environment parameters or directly intervening with a learned adversary during data collection. In many real-world applications, however, we do not have access to fast, high-fidelity simulators for training (Cao et al., 2022; Mowlavi et al., 2022; Xu et al., 2023). In these cases,

we must be able to incorporate robustness to model uncertainty without relying on multiple training environments or potentially dangerous adversarial interventions, as real-world data collection may be necessary.

In Chapter 5 and Chapter 6 of this dissertation, we introduce techniques that provide robustness to model uncertainty without introducing additional assumptions on data collection during training. Therefore, these methods are compatible with settings that require real-world interaction for training, and can be broadly applied to any setting where we can collect data from a single training environment.

1.3 Contributions and Outline

This dissertation contributes new algorithms for *reliable deep RL* that address important problems for real-world decision making and control. First, Chapter 2 provides background on the key ideas addressed in the dissertation. Then, Chapter 3 and Chapter 4 consider the requirement of *stable training from limited data*, and introduce policy improvement algorithms with improved data efficiency. Next, Chapter 5 and Chapter 6 consider the need for *robust, safe deployment in the presence of uncertainty*, and introduce techniques that provide robustness to model uncertainty while only requiring standard data collection in a single training environment. Finally, Chapter 7 concludes the dissertation with a discussion of future research directions for the development of reliable deep RL algorithms.

These contributions are motivated by the need to overcome key barriers preventing the adoption of deep RL in important application areas, which we accomplish through principled algorithm design. Overall, the methods introduced in this dissertation provide a foundation for the future development and deployment of reliable deep RL algorithms in areas of societal importance. We now describe the detailed contributions of each chapter in the dissertation.

In Chapter 3, we address the training instability caused by sample-based uncertainty in policy improvement algorithms. We develop the algorithm *Uncertainty-Aware Trust Region Policy Optimization (UA-TRPO)* that controls the finite-sample estimation error in the main components of the popular deep RL algorithm TRPO (Schulman et al., 2015). UA-TRPO is theoretically supported by a novel finite-sample policy improvement lower bound, which we use to motivate an adaptive trust region that directly considers the uncertainty in the surrogate objective estimate. We also propose a technique that restricts policy updates to a subspace where trust region information is available from the observed data. This chapter is based on the work in Queeney et al. (2021b).

In Chapter 4, we improve the data efficiency of policy improvement algorithms through sample reuse. We develop a class of *Generalized Policy Improvement (GPI)* algorithms that extends on-policy methods to incorporate sample reuse, without sacrificing their approximate policy improvement guarantees during training. These algorithms are constructed based on a novel Generalized Policy Improvement lower bound that we introduce, which is compatible with the use of data from past policies. Using this lower bound, we demonstrate how to optimally reuse data from all recent policies. Our *theoretically supported sample reuse* improves the trade-off between batch size and policy update size throughout training compared to on-policy methods, while retaining the same approximate policy improvement guarantees. This chapter is based on the work in Queeney et al. (2021a) and Queeney et al. (2022).

In Chapter 5, we introduce a framework to guarantee robust, safe deployment in the presence of model uncertainty through the use of an optimal transport cost uncertainty set. We show that worst-case optimization problems over this uncertainty set of transition models can be reformulated as adversarial perturbations to state transitions in the training environment. This motivates an efficient, model-free implementation

based on applying *Optimal Transport Perturbations (OTP)* to construct worst-case *virtual* state transitions, which does not impact data collection during training and does not require detailed simulator access. These perturbations can be added to the training process of any safe RL algorithm to incorporate robustness to general disturbances at deployment time, while only requiring standard data collection in a single training environment. This chapter is based on the work in Queeney et al. (2023).

In Chapter 6, we introduce a risk-averse perspective towards model uncertainty to provide robust, safe deployment of learned control policies. We consider a distribution over transition models, and reformulate deep RL with safety constraints to incorporate *Risk-Averse Model Uncertainty (RAMU)* through the use of coherent distortion risk measures. From a theoretical standpoint, we provide robustness guarantees for the RAMU framework by showing it is equivalent to a specific class of distributionally robust safe RL problems. Unlike existing approaches to robustness in deep RL, however, this formulation does not involve minimax optimization. This leads to an efficient, model-free implementation using weighted sample averages that only requires data collected from a single training environment. This chapter is based on the work in Queeney and Benosman (2023).

Throughout the dissertation, experiments on a variety of continuous control tasks demonstrate the effectiveness of our algorithms. We include experiments on OpenAI Gym's MuJoCo environments (Brockman et al., 2016; Todorov et al., 2012), tasks from the DeepMind Control Suite (Tunyasuvunakool et al., 2020), and tasks with safety constraints from the Real-World RL Suite (Dulac-Arnold et al., 2020, 2021). For reproducibility and future research, we provide publicly available implementations of all algorithms introduced in this dissertation.[1]

[1]Code is publicly available at `https://github.com/jqueeney`.

Chapter 2

Background

Before introducing new algorithms for reliable deep RL, we provide background on the key ideas explored throughout the dissertation. We include a literature review of related work, and introduce the problem formulations used in the remainder of the dissertation.

2.1 Related Work: Stable Training from Limited Data

The first part of the dissertation focuses on the goal of *stable training from limited data*, and extends existing policy improvement algorithms to account for the error introduced by (i) sample-based uncertainty and (ii) sample reuse.

2.1.1 On-Policy Policy Improvement Methods

The goal of monotonic policy improvement was first introduced by Kakade and Langford (2002) in Conservative Policy Iteration, which considers a mixture between the current and greedy policies at every update. The policy improvement bounds proposed by Kakade and Langford (2002) were later refined by Schulman et al. (2015) and Achiam et al. (2017), making them compatible with the deep RL setting. This theory of policy improvement has served as a fundamental building block in the design of on-policy deep RL methods, including the popular algorithms Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) and Proximal Policy Optimization (PPO) (Schulman et al., 2017). In addition, on-policy methods based on non-parametric tar-

get policies, such as On-Policy Maximum a Posteriori Policy Optimization (VMPO) (Song et al., 2020), can also be interpreted from a policy improvement perspective.

Unfortunately, existing policy improvement methods do not provide practical performance guarantees in real-world settings where data collection is difficult, as they require the use of on-policy data and only hold in expectation. In these cases, we must address the error introduced from limited data in order to guarantee stable training in practice.

2.1.2 Sample-Based Uncertainty from Limited Data

Deep RL algorithms rely on sample-based estimates to approximate expectations, which introduce sample-based uncertainty into policy updates. These estimates are known to suffer from high variance, so variance reduction techniques have been proposed to mitigate this issue (Sutton et al., 2000; Schulman et al., 2016; Papini et al., 2018). However, when access to data is limited, significant sample-based uncertainty may remain. This source of error has also been referred to as *epistemic uncertainty*.

Li et al. (2011) developed the "knows what it knows" (KWIK) framework for this scenario, which allows an algorithm to choose not to produce an output when uncertainty is high. Several approaches in RL can be viewed as applications of the KWIK framework. Laroche et al. (2019) bootstrapped the learned policy with a known baseline policy in areas of the state space where data was limited, while Thomas et al. (2015) only produced updates when the value of the new policy exceeded a baseline value with high probability. A related line of work directly accounts for epistemic uncertainty by learning conservative estimates of the value function (Kumar et al., 2020) or transition model (Kidambi et al., 2020), which are then used to guide policy updates. The policy improvement method that we introduce in Chapter 3 also applies conservative, uncertainty-aware estimates at every update in order to guarantee stable training from limited data.

2.1.3 Data Efficiency with Sample Reuse

A common approach to improving the data efficiency of deep RL is to reuse samples collected under prior policies. Off-policy deep RL algorithms (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018; Abdolmaleki et al., 2018) achieve data efficiency through the use of a replay buffer during training, which allows samples to be used for multiple policy updates. Typically, the replay buffer stores millions of samples, which enables state-of-the-art performance on popular benchmarks. However, this aggressive form of sample reuse can lead to high computation and memory requirements when combined with deep neural network representations and rich sensory data such as images. As a result, popular off-policy algorithms are not compatible with many resource-constrained control settings that require real-time, on-device learning (Jang et al., 2020; Duisterhof et al., 2021; Grossman and Plancher, 2022; Neuman et al., 2022; Pau et al., 2023). On-policy approaches, on the other hand, require significantly less computation and memory compared to off-policy algorithms (Grossman and Plancher, 2022), making them a viable option in settings that preclude the use of large replay buffers.

Most importantly, the characteristics of the replay buffer are treated as hyperparameters in off-policy deep RL, resulting in aggressive, resource-intensive sample reuse that lacks practical performance guarantees. Methods have been proposed to address the bias introduced by off-policy data. One line of work combines on-policy and off-policy updates to address this concern (O'Donoghue et al., 2017; Gu et al., 2017a,b; Wang et al., 2017; Fakoor et al., 2020; Meng et al., 2022), and some of these methods consider penalty terms that account for the difference between the current policy and the data-generating policy. Other approaches have modified the use of the replay buffer to control the bias from off-policy data. Novati and Koumoutsakos (2019) ignored samples from the replay buffer whose actions were not likely

under the current policy, and Wang et al. (2020) considered a sampling scheme that emphasized more recent experience in the replay buffer. The Generalized Policy Improvement lower bound that we introduce in Chapter 4 motivates similar penalty terms and non-uniform weighting of samples, which guarantees stable training while reusing samples for improved data efficiency.

2.2 Related Work: Robust and Safe Deployment

The second part of the dissertation focuses on the need for *robust, safe deployment in the presence of uncertainty*, and builds upon standard representations of safety and model uncertainty in deep RL.

2.2.1 Safety Constraints

The most common approach to modeling safety in RL is to incorporate constraints on expected total costs, which is the definition of safety we consider in this dissertation. In recent years, several deep RL algorithms have been developed for this framework. A popular approach is to solve the Lagrangian relaxation of the constrained problem (Tessler et al., 2019b; Ray et al., 2019; Stooke et al., 2020), which is supported by theoretical results establishing that constrained RL has zero duality gap (Paternain et al., 2019). Other approaches to safe RL construct closed-form solutions to guide policy updates (Achiam et al., 2017; Liu et al., 2022), or consider immediate switching between the objective and constraints to better satisfy safety during training (Xu et al., 2021). However, these safe RL algorithms are only designed to satisfy expected cost constraints in the training environment.

2.2.2 Model Uncertainty

Recall that *model uncertainty* represents irreducible uncertainty about the true environment at deployment time, and is distinct from epistemic uncertainty that can be

reduced through additional data collection. We can account for model uncertainty by considering a range of possible environments during training. Existing deep RL methods accomplish this by applying an uncertainty set or a distribution over transition models.

Robust RL methods account for uncertainty in the environment by considering worst-case transition models from an uncertainty set (Nilim and Ghaoui, 2005; Iyengar, 2005). In order to facilitate efficient implementations in the deep RL setting, most techniques have focused on adversarial interventions or parametric uncertainty. Adversarial methods consider direct intervention with a learned adversary during trajectory rollouts (Pinto et al., 2017; Tessler et al., 2019a; Vinitsky et al., 2020), and these adversarial perturbations are trained to minimize performance. Parametric approaches, on the other hand, consider robustness with respect to environment characteristics that can be altered in a simulator (Rajeswaran et al., 2017; Mankowitz et al., 2020, 2021).

Model uncertainty can also be represented by a distribution over potential environments. Domain randomization (Tobin et al., 2017; Peng et al., 2018) is a parametric uncertainty method that randomizes across parameter values in a simulator, and trains a policy to maximize average performance over this training distribution. This represents a risk-neutral attitude towards model uncertainty that works well in practice but lacks robustness guarantees, so it has been referred to as a soft-robust approach (Derman et al., 2018). Distributionally robust RL methods incorporate robustness to the choice of training distribution by instead considering an ambiguity set of distributions (Xu and Mannor, 2010; Yu and Xu, 2016; Derman and Mannor, 2020; Chen and Paschalidis, 2020), but application of this distributionally robust framework has remained limited in deep RL as it requires solving for worst-case distributions over transition models.

A major drawback of existing approaches is their need to directly modify the environment during training. Parametric methods assume the ability to generate a range of training environments with a detailed simulator, while adversarial methods directly influence the data collection process by attempting to negatively impact performance. The techniques that we introduce in Chapter 5 and Chapter 6, on the other hand, do not require detailed simulator access and do not impact the data collection process during training.

2.3 Reinforcement Learning Framework

Throughout the dissertation, we represent the sequential decision making problem as an infinite-horizon, discounted Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, d_0, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$ is the transition probability function where $P(\mathcal{S})$ denotes the space of probability measures over $\mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, d_0 is the initial state distribution, and γ is the discount rate.

We model the agent's decisions as a stationary policy $\pi : \mathcal{S} \to P(\mathcal{A})$, which outputs a distribution of actions at every state. Our goal is to find a policy π that maximizes the expected total discounted rewards

$$J(\pi) = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],$$

where $\tau \sim \pi$ represents a trajectory sampled according to $s_0 \sim d_0$, $a_t \sim \pi(\cdot \mid s_t)$, and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$. A policy π also induces a normalized discounted state visitation distribution d^π, where $d^\pi(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s \mid d_0, \pi, p)$. We write the corresponding normalized discounted state-action visitation distribution as $d^\pi(s, a) = d^\pi(s)\pi(a \mid s)$, where we make it clear from the context whether d^π refers to a distribution over states or state-action pairs.

We denote the state value function of π as $V^\pi(s) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t) \mid s_0 = s\right]$, the state-action value function as $Q^\pi(s,a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t) \mid s_0 = s, a_0 = a\right]$ which is commonly referred to as the Q function, and the advantage function as $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. In the context of deep RL, the policy and value functions are parameterized by neural networks.

2.4 Preliminaries: Stable Training from Limited Data

In order to guarantee stable training from limited data, Chapter 3 and Chapter 4 develop policy improvement algorithms with a focus on data efficiency. These algorithms are motivated by the design of existing on-policy policy improvement algorithms, which we describe next.

2.4.1 On-Policy Policy Improvement Lower Bound

Many popular on-policy algorithms can be interpreted as approximately maximizing the following policy improvement lower bound, which was first developed by Kakade and Langford (2002) and later refined by Schulman et al. (2015) and Achiam et al. (2017).

Lemma 2.1 (Achiam et al. 2017)**.** *Consider any policy π and a current policy π_k. Then,*

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}}\left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)}A^{\pi_k}(s,a)\right] - \frac{2\gamma C^{\pi,\pi_k}}{(1-\gamma)^2}\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\mathrm{TV}(\pi,\pi_k)(s)\right], \tag{2.1}$$

where $C^{\pi,\pi_k} = \max_{s\in\mathcal{S}}\left|\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A^{\pi_k}(s,a)\right]\right|$ and $\mathrm{TV}(\pi,\pi_k)(s)$ represents the total variation (TV) distance between the distributions $\pi(\cdot \mid s)$ and $\pi_k(\cdot \mid s)$.

We refer to the first term of the lower bound in Lemma 2.1 as the surrogate objective and the second term as the penalty term. Note that we can guarantee monotonic policy improvement at every step of the learning process by choosing the next policy π_{k+1}

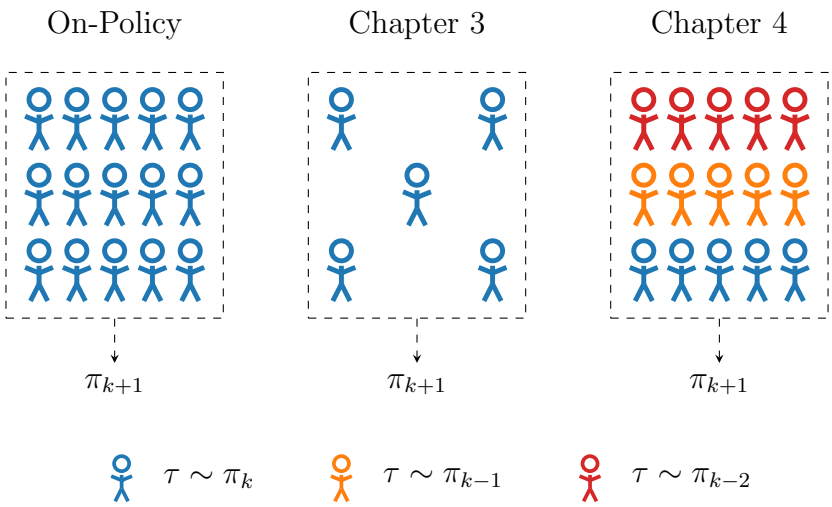

Figure 2·1: Comparison of updates in policy improvement algorithms. Left: On-policy methods require a large number of samples from the current policy π_k. Center: Chapter 3 addresses sample-based uncertainty in the limited data setting. Right: Chapter 4 improves data efficiency by reusing samples collected under prior policies.

to maximize this lower bound, leading to reliable performance throughout training. However, the expectations that appear in the surrogate objective and penalty term must be estimated using samples collected under the current policy π_k. Therefore, this bound motivates algorithms that are practical to implement, but may result in slow learning due to their requirement of on-policy data and the need for large sample sizes to approximate expectations. See Figure 2·1 for an illustration of these requirements, including a comparison to the settings we consider in Chapter 3 and Chapter 4 of the dissertation.

2.4.2 On-Policy Policy Improvement Algorithms

Rather than directly maximize the lower bound in Lemma 2.1, on-policy policy improvement algorithms typically maximize the surrogate objective while bounding the risk of each policy update via a constraint on the penalty term. This leads to updates with the following form.

Algorithm 2.1: On-Policy Policy Improvement Algorithms

Input: initial policy π_0; TV distance trust region parameter ϵ; batch size N.

for $k = 0, 1, 2, \ldots$ **do**

Collect N samples with π_k.

Use these N samples to approximate the expectations in Definition 2.1.

Update policy by approximately solving the optimization problem in Definition 2.1. Implementation varies by algorithm.

end

Definition 2.1 (On-Policy Trust Region Update)**.** *For a given choice of trust region parameter $\epsilon > 0$, the* on-policy trust region update *has the form*

$$\max_{\pi} \underset{(s,a)\sim d^{\pi_k}}{\mathbb{E}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a) \right] \quad \text{s.t.} \quad \underset{s\sim d^{\pi_k}}{\mathbb{E}} \left[\mathrm{TV}\left(\pi, \pi_k\right)(s) \right] \leq \frac{\epsilon}{2}. \tag{2.2}$$

The trust region in (2.2) bounds the magnitude of the penalty term in (2.1), limiting the worst-case performance decline at every update. Therefore, we say that on-policy algorithms based on the trust region update in Definition 2.1 deliver *approximate* policy improvement guarantees. In addition, practical deep RL implementations of this update introduce additional approximations through the use of sample-based estimates.

The high-level framework of on-policy policy improvement algorithms is described in Algorithm 2.1. The difference between popular on-policy algorithms is primarily due to how they approximately solve the optimization problem in Definition 2.1. PPO applies a TV distance trust region via a clipping mechanism, while TRPO and VMPO instead consider forward and reverse Kullback-Leibler (KL) divergence trust regions, respectively. These related trust regions guarantee that the TV distance trust region in (2.2) is satisfied, as shown in the following result.

Lemma 2.2. *Consider a current policy π_k, and any policy π that satisfies*

$$\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\mathrm{KL}\left(\pi_k\|\pi\right)(s)\right] \leq \delta, \tag{2.3}$$

or

$$\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\mathrm{KL}\left(\pi\|\pi_k\right)(s)\right] \leq \delta, \tag{2.4}$$

where $\mathrm{KL}\left(\pi_k\|\pi\right)(s)$ *and* $\mathrm{KL}\left(\pi\|\pi_k\right)(s)$ *represent the forward and reverse KL divergence of the distribution $\pi(\cdot \mid s)$ from the distribution $\pi_k(\cdot \mid s)$, respectively, and $\delta = \epsilon^2/2$. Then, π also satisfies the TV distance trust region in* (2.2).

Proof. Note that TV distance is symmetric, so the same argument can be used starting from (2.3) or (2.4). By applying Pinsker's inequality followed by Jensen's inequality, we see that

$$\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\mathrm{TV}\left(\pi,\pi_k\right)(s)\right] \leq \sqrt{\frac{1}{2}\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\mathrm{KL}\left(\pi\|\pi_k\right)(s)\right]} \leq \sqrt{\frac{\delta}{2}} = \frac{\epsilon}{2}.$$

□

2.5 Preliminaries: Robust and Safe Deployment

In order to guarantee reliable performance and safety upon deployment, Chapter 5 and Chapter 6 incorporate two additional components into the standard RL framework: (i) safety constraints and (ii) model uncertainty.

2.5.1 Safety Constraints

We incorporate safety constraints through the use of a Constrained MDP (C-MDP) (Altman, 1999) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, c, d_0, \gamma)$, where $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the cost function used to define the safety constraint. We focus on the setting with a single constraint, but all results in the dissertation can be extended to the case of multiple constraints.

In the C-MDP setting, we write the expected total discounted rewards as $J_{p,r}(\pi)$ and the corresponding reward Q function as $Q^{\pi}_{p,r}(s,a)$, where we include subscripts to

denote the transition model p and reward function r. Similarly, we write the expected total discounted costs as $J_{p,c}(\pi)$ and the corresponding cost Q function as $Q^{\pi}_{p,c}(s,a)$. For a given C-MDP, the goal of safe RL is to find a policy π that maximizes the constrained optimization problem

$$\max_{\pi} \; J_{p,r}(\pi) \quad \text{s.t.} \quad J_{p,c}(\pi) \leq B, \tag{2.5}$$

where B is a safety budget on expected total discounted costs. Off-policy optimization techniques (Xu et al., 2021; Liu et al., 2022) iteratively optimize (2.5) by considering the related optimization problem

$$\max_{\pi} \; \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[Q^{\pi_k}_{p,r}(s,a)\right]\right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[Q^{\pi_k}_{p,c}(s,a)\right]\right] \leq B, \tag{2.6}$$

where π_k is the current policy and $\mathcal{D}$ is a replay buffer containing data collected during training.

2.5.2 Model Uncertainty

Rather than focusing on a single C-MDP with transition model p, we can incorporate uncertainty about the transition model into the constrained optimization problem in (2.5). The two most common approaches for representing model uncertainty apply either an uncertainty set or a distribution over transition models. See Figure 2·2 for an illustration. We consider both approaches in this dissertation.

In Chapter 5, we consider an uncertainty set $\mathcal{P}$ of transition models. We assume $\mathcal{P} = \bigotimes_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{P}_{s,a}$, where $\mathcal{P}_{s,a}$ is a set of transition models $p_{s,a} = p(\,\cdot \mid s,a) \in P(\mathcal{S})$ at a given state-action pair and $\mathcal{P}$ is the product of these sets. We apply the Robust Constrained MDP (RC-MDP) framework (Mankowitz et al., 2021; Russel et al., 2021),

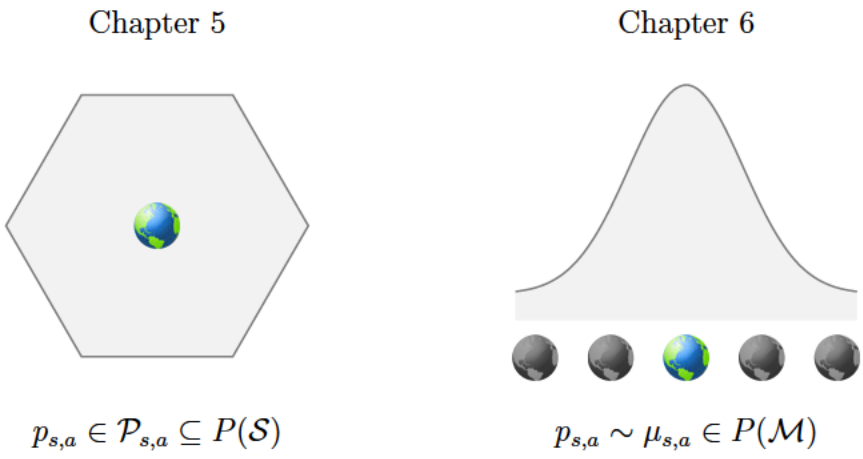

Figure 2·2: Popular representations of model uncertainty in deep RL. Left: Uncertainty set $\mathcal{P}_{s,a} \subseteq P(\mathcal{S})$ over transition models at a given state-action pair, where $p_{s,a} \in \mathcal{P}_{s,a}$. Right: Distribution $\mu_{s,a} \in P(\mathcal{M})$ over transition models at a given state-action pair, where $p_{s,a} \sim \mu_{s,a}$.

which considers a robust version of (2.5) given by

$$\max_{\pi} \inf_{p \in \mathcal{P}} J_{p,r}(\pi) \quad \text{s.t.} \quad \sup_{p \in \mathcal{P}} J_{p,c}(\pi) \leq B. \tag{2.7}$$

In Chapter 6, we instead consider a distribution μ over transition models. We focus on distributions of the form $\mu = \prod_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mu_{s,a}$, where $\mu_{s,a}$ represents a distribution over transition models $p_{s,a} \in P(\mathcal{S})$ at a given state-action pair and μ is the product over all $\mu_{s,a}$. Note that $\mu_{s,a} \in P(\mathcal{M})$, where we write $\mathcal{M} = P(\mathcal{S})$ to denote model space. It is common to apply the expectation operator over μ (Derman et al., 2018; Peng et al., 2018), leading to the constrained optimization problem

$$\max_{\pi} \mathop{\mathbb{E}}_{p \sim \mu} [J_{p,r}(\pi)] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{p \sim \mu} [J_{p,c}(\pi)] \leq B. \tag{2.8}$$

Distributionally robust MDPs (Xu and Mannor, 2010; Yu and Xu, 2016) incorporate robustness to the choice of μ by considering an ambiguity set $\mathcal{U} = \bigotimes_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mathcal{U}_{s,a}$ of distributions over transition models, where $\mu_{s,a} \in \mathcal{U}_{s,a} \subseteq P(\mathcal{M})$. This results in the

distributionally robust safe RL problem

$$\max_{\pi} \inf_{\mu \in \mathcal{U}} \mathop{\mathbb{E}}_{p \sim \mu} [J_{p,r}(\pi)] \quad \text{s.t.} \quad \sup_{\mu \in \mathcal{U}} \mathop{\mathbb{E}}_{p \sim \mu} [J_{p,c}(\pi)] \leq B. \tag{2.9}$$

We show in Chapter 6 that our Risk-Averse Model Uncertainty framework is equivalent to (2.9) for appropriate choices of ambiguity sets in the objective and constraint.

Finally, note that the product structure of $\mathcal{P}$, μ, and $\mathcal{U}$ over state-action pairs is known as rectangularity, and is a common assumption in the literature (Nilim and Ghaoui, 2005; Iyengar, 2005; Xu and Mannor, 2010; Yu and Xu, 2016; Derman et al., 2018; Derman and Mannor, 2020; Chen and Paschalidis, 2020).

Chapter 3

Uncertainty-Aware Trust Region Policy Optimization

Given the data-driven nature of deep RL, we must rely on sample-based estimates to approximate expectations. Unfortunately, these estimates are known to suffer from high variance, particularly for problems with long time horizons. As a result, finite-sample estimation error can be a major source of training instability in deep RL algorithms when small sample sizes are used, which is often the case in real-world decision making settings. Notably, sample-based uncertainty can destroy the stable training benefits of policy improvement methods, since the lower bound in Lemma 2.1 depends on expectations in both the surrogate objective and penalty term. In this chapter, we extend these policy improvement guarantees to the finite-sample setting that applies in practice. In particular, we develop methods to control the sample-based uncertainty present in Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), resulting in an uncertainty-aware algorithm that makes robust use of limited data to deliver stable performance throughout the training process.

Throughout this chapter, we use bold lowercase letters to denote vectors, bold uppercase letters to denote matrices, and hats ($\hat{\cdot}$) to denote sample-based estimates.

3.1 Trust Region Policy Optimization

TRPO is an on-policy policy improvement algorithm that approximates the update in Definition 2.1 by considering the forward KL divergence trust region in (2.3). This

leads to the policy update

$$\max_{\pi} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a) \right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{KL}\left(\pi_k \| \pi\right)(s) \right] \leq \delta. \tag{3.1}$$

Next, TRPO introduces several modifications to produce a scalable and practical algorithm. Specifically, TRPO considers a first order approximation of the surrogate objective and a second order approximation of the KL divergence trust region in (3.1). For neural network policies π and π_k parameterized by $\boldsymbol{\theta}, \boldsymbol{\theta}_k \in \mathbb{R}^d$, respectively, these approximations are given by

$$\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a) \right] \approx \mathbf{g}_k' \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right), \tag{3.2}$$

$$\mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{KL}\left(\pi_k \| \pi\right)(s) \right] \approx \frac{1}{2} \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right)' \mathbf{F}_k \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right), \tag{3.3}$$

with

$$\mathbf{g}_k = \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[A^{\pi_k}(s,a) \nabla_{\boldsymbol{\theta}} \log \pi_k(a \mid s) \right], \tag{3.4}$$

$$\mathbf{F}_k = \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_k(a \mid s) \nabla_{\boldsymbol{\theta}} \log \pi_k(a \mid s)' \right], \tag{3.5}$$

where $\mathbf{g}_k$ is the standard policy gradient determined by the Policy Gradient Theorem (Williams, 1992; Sutton et al., 2000) and $\mathbf{F}_k$ is the average Fisher Information Matrix (Schulman et al., 2015). By applying these approximations, we see that the TRPO update is alternatively motivated by the following approximate policy improvement lower bound.

Corollary 3.1. *Consider a current policy π_k parameterized by $\boldsymbol{\theta}_k \in \mathbb{R}^d$, and any policy π parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$. Let C^{π,π_k} be defined as in Lemma 2.1. Then,*

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma} \mathbf{g}_k' \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right) - \frac{\gamma C^{\pi,\pi_k}}{(1-\gamma)^2} \sqrt{\left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right)' \mathbf{F}_k \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right)},$$

up to first and second order approximation error.

Proof. Starting from the lower bound in Lemma 2.1, we can apply the same techniques as in Lemma 2.2 to bound the TV distance penalty term with a forward KL divergence penalty term. Then, by using the surrogate objective approximation in (3.2) and the forward KL divergence approximation in (3.3), we obtain the result. □

Note that $\mathbf{g}_k$ and $\mathbf{F}_k$ are themselves expectations, so in practice they are estimated using sample averages $\hat{\mathbf{g}}_k$ and $\hat{\mathbf{F}}_k$. This results in the optimization problem

$$\max_{\boldsymbol{\theta}} \ \hat{\mathbf{g}}_k' \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k\right) \quad \text{s.t.} \quad \frac{1}{2} \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k\right)' \hat{\mathbf{F}}_k \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k\right) \leq \delta, \tag{3.6}$$

which we use to determine the parameterization $\boldsymbol{\theta}_{k+1}$ of the updated policy π_{k+1}. The closed-form solution of (3.6) is given by $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \mathbf{v}$, where $\mathbf{v} = \hat{\mathbf{F}}_k^{-1} \hat{\mathbf{g}}_k$ is the update direction and $\eta = \sqrt{2\delta / \mathbf{v}' \hat{\mathbf{F}}_k \mathbf{v}}$. The update direction cannot be calculated directly in high dimensions, so it is solved approximately by applying a finite number of conjugate gradient steps to $\hat{\mathbf{F}}_k \mathbf{v} = \hat{\mathbf{g}}_k$. Finally, a backtracking line search is performed to account for the error introduced by the first and second order approximations.

By replacing expectations $\mathbf{g}_k$ and $\mathbf{F}_k$ with sample-based estimates $\hat{\mathbf{g}}_k$ and $\hat{\mathbf{F}}_k$, TRPO introduces a potentially significant source of error when the number of samples used to construct the estimates is small. This finite-sample estimation error can destroy the approximate policy improvement argument on which TRPO is based. As a result, TRPO typically uses large amounts of data to generate stable performance. However, this approach is often not practical in real-world decision making settings.

3.2 Uncertainty-Aware Subspace

We first address the error present in the trust region estimate in (3.6). Because we are estimating a high-dimensional matrix using a limited number of samples, $\hat{\mathbf{F}}_k$ is unlikely to be full rank. This creates multiple problems when approximating the update direction $\mathbf{v} = \mathbf{F}_k^{-1} \mathbf{g}_k$ by solving the system of equations $\hat{\mathbf{F}}_k \mathbf{v} = \hat{\mathbf{g}}_k$. First,

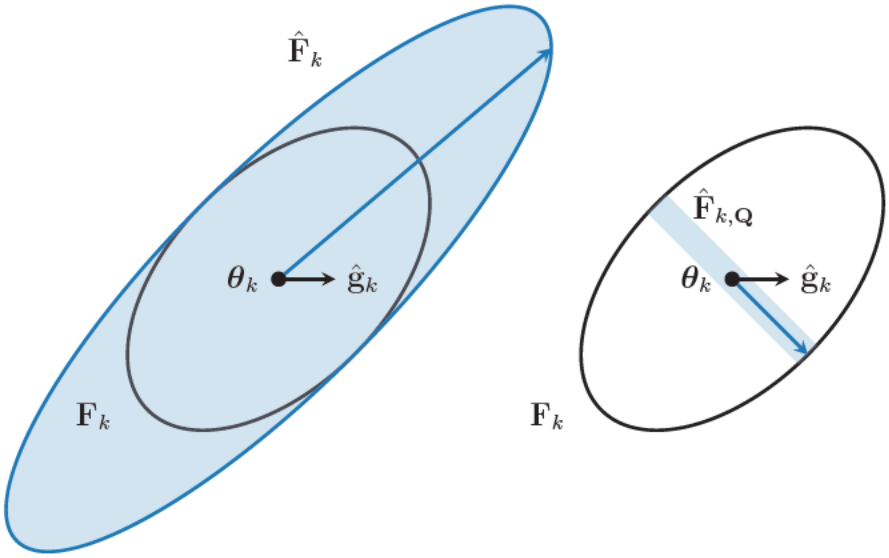

Figure 3·1: Impact of restricting trust region estimate to uncertainty-aware subspace. True trust region in black. Estimated trust region and corresponding policy update in blue. Left: Finite-sample estimation error in the trust region estimate can lead to policy updates that are outside of the true trust region. Right: By restricting updates to a subspace where trust region information is available, the resulting policy update remains within the true trust region.

it is unlikely that this system of equations has an exact solution, so we must consider a least-squares solution instead. Second, the least-squares solution is not unique because the null space of $\hat{\mathbf{F}}_k$ contains non-zero directions. This second point is particularly important for managing uncertainty, as the null space of $\hat{\mathbf{F}}_k$ can be interpreted as the directions in parameter space where we have no information on the trust region metric from observed data. Even with the addition of a damping coefficient for regularization, this lack of information can lead to large policy updates that are prone to instability as shown on the left-hand side of Figure 3·1.

In order to produce a stable, uncertainty-aware policy update, we should restrict our attention to directions in parameter space where an estimate of the trust region metric is available. Mathematically, this means we are interested in finding a least-squares solution to $\hat{\mathbf{F}}_k\mathbf{v} = \hat{\mathbf{g}}_k$ that is contained in the row space of $\hat{\mathbf{F}}_k$ (equivalently,

Algorithm 3.1: Basis for Uncertainty-Aware Subspace

Input: sample-based estimate $\hat{\mathbf{F}}_k \in \mathbb{R}^{d\times d}$; random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{d\times m}$.

Generate m random projections onto the range of $\hat{\mathbf{F}}_k$:

$$\mathbf{Y} = \hat{\mathbf{F}}_k \boldsymbol{\Omega}.$$

Construct basis $\mathbf{Q} \in \mathbb{R}^{d\times \ell}$, $\ell \leq m$, with orthonormal vectors via the singular value decomposition of $\mathbf{Y}$.

the range of $\hat{\mathbf{F}}_k$ since the matrix is symmetric). The unique least-squares solution that satisfies this restriction is $\mathbf{v} = \hat{\mathbf{F}}_k^{+}\hat{\mathbf{g}}_k$, where $\hat{\mathbf{F}}_k^{+}$ denotes the Moore-Penrose pseudoinverse of $\hat{\mathbf{F}}_k$. It is important to note that the standard implementation of TRPO does not produce this update direction in general, which can lead to unstable and inefficient updates when sample sizes are small.

If $\hat{\mathbf{F}}_k$ has rank $p < d$, it can be written as $\hat{\mathbf{F}}_k = \mathbf{U}\mathbf{D}\mathbf{U}'$ where $\mathbf{U} \in \mathbb{R}^{d\times p}$ is an orthonormal eigenbasis for the range of $\hat{\mathbf{F}}_k$ and $\mathbf{D} \in \mathbb{R}^{p\times p}$ is a diagonal matrix of the corresponding positive eigenvalues. The Moore-Penrose pseudoinverse is $\hat{\mathbf{F}}_k^{+} = \mathbf{U}\mathbf{D}^{-1}\mathbf{U}'$, and the update direction can be calculated as $\mathbf{v} = \mathbf{U}\mathbf{D}^{-1}\mathbf{U}'\hat{\mathbf{g}}_k$. Intuitively, this solution is obtained by first restricting $\hat{\mathbf{F}}_k$ and $\hat{\mathbf{g}}_k$ to the p-dimensional subspace spanned by the basis $\mathbf{U}$, finding the unique solution to the resulting p-dimensional system of equations, and representing this solution in terms of its coordinates in parameter space.

Unfortunately, standard methods for computing a decomposition of $\hat{\mathbf{F}}_k$ are computationally intractable in high dimensions. Rather than considering the full range of $\hat{\mathbf{F}}_k$ spanned by $\mathbf{U}$, we propose to instead restrict policy updates to a low-rank subspace of the range using random projections (Halko et al., 2011). By generating $m \ll d$ random projections, we can efficiently calculate a basis $\mathbf{Q} \in \mathbb{R}^{d\times \ell}$, $\ell \leq m$, for this uncertainty-aware subspace as detailed in Algorithm 3.1. This basis can be used

to construct the corresponding uncertainty-aware projections

$$\hat{\mathbf{F}}_{k,\mathbf{Q}} = \mathbf{Q}'\hat{\mathbf{F}}_k\mathbf{Q} \in \mathbb{R}^{\ell\times\ell}, \quad \hat{\mathbf{g}}_{k,\mathbf{Q}} = \mathbf{Q}'\hat{\mathbf{g}}_k \in \mathbb{R}^{\ell},$$

where we use a subscript $\mathbf{Q}$ to denote projections in the coordinates given by this basis. This leads to the low-dimensional policy update

$$\max_{\mathbf{y}} \ \hat{\mathbf{g}}'_{k,\mathbf{Q}}\mathbf{y} \quad \text{s.t.} \quad \frac{1}{2}\mathbf{y}'\hat{\mathbf{F}}_{k,\mathbf{Q}}\mathbf{y} \leq \delta \tag{3.7}$$

in the subspace spanned by the basis $\mathbf{Q}$, where $\boldsymbol{\theta} = \boldsymbol{\theta}_k + \mathbf{Q}\mathbf{y}$. Because the subspace is contained in the range of $\hat{\mathbf{F}}_k$ by construction, we preserve our original goal of restricting policy updates to directions in parameter space where trust region information is available as shown on the right-hand side of Figure 3·1.

Finally, for updates restricted to the uncertainty-aware subspace, we can rewrite the approximate policy improvement lower bound in Corollary 3.1 as a function of the low-dimensional policy update $\mathbf{y} \in \mathbb{R}^{\ell}$. For a policy π parameterized by $\boldsymbol{\theta} = \boldsymbol{\theta}_k + \mathbf{Q}\mathbf{y}$, up to first and second order approximation error we have that

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma}\mathbf{g}'_{k,\mathbf{Q}}\mathbf{y} - \frac{\gamma C^{\pi,\pi_k}}{(1-\gamma)^2}\sqrt{\mathbf{y}'\mathbf{F}_{k,\mathbf{Q}}\mathbf{y}}, \tag{3.8}$$

where $\mathbf{g}_{k,\mathbf{Q}} = \mathbf{Q}'\mathbf{g}_k \in \mathbb{R}^{\ell}$ and $\mathbf{F}_{k,\mathbf{Q}} = \mathbf{Q}'\mathbf{F}_k\mathbf{Q} \in \mathbb{R}^{\ell\times\ell}$ are projections of $\mathbf{g}_k$ and $\mathbf{F}_k$, respectively, onto the subspace spanned by $\mathbf{Q}$.

3.3 Finite-Sample Policy Improvement Lower Bound

Although we have restricted updates to an uncertainty-aware subspace, the sample-based estimate of the projected policy gradient in (3.7) introduces another source of potential error. Rather than relying on the high-variance estimate $\hat{\mathbf{g}}_{k,\mathbf{Q}}$ to approximate $\mathbf{g}_{k,\mathbf{Q}}$ in (3.8), we instead develop a robust, uncertainty-aware lower bound that holds for all vectors in an uncertainty set $\mathcal{U}_k$ centered around $\hat{\mathbf{g}}_{k,\mathbf{Q}}$. If $\mathcal{U}_k$ contains the

true projected policy gradient $\mathbf{g}_{k,\mathbf{Q}}$, this will be a lower bound to $J(\pi) - J(\pi_k)$ up to first and second order approximation error.

Consider the policy gradient random vector

$$\boldsymbol{\xi}_k = A^{\pi_k}(s,a)\nabla_{\boldsymbol{\theta}} \log \pi_k(a \mid s) \in \mathbb{R}^d,$$

where $(s,a) \sim d^{\pi_k}$ and $\mathbf{g}_k = \mathbb{E}\left[\boldsymbol{\xi}_k\right]$ is the true policy gradient as in (3.4). Because we are interested in updates restricted to the uncertainty-aware subspace spanned by $\mathbf{Q}$, we focus on the projected policy gradient random vector $\boldsymbol{\xi}_{k,\mathbf{Q}} = \mathbf{Q}'\boldsymbol{\xi}_k \in \mathbb{R}^\ell$. Note that $\mathbf{g}_{k,\mathbf{Q}} = \mathbb{E}\left[\boldsymbol{\xi}_{k,\mathbf{Q}}\right]$ is the true projected policy gradient, and $\boldsymbol{\Sigma}_{k,\mathbf{Q}} = \mathbb{E}\left[\left(\boldsymbol{\xi}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}}\right)\left(\boldsymbol{\xi}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}}\right)'\right]$ is the true covariance matrix of the projected policy gradient random vector. We make the following assumption regarding the standardized projected random vector $\boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1/2}(\boldsymbol{\xi}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}})$.

Assumption 3.1. *$\boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1/2}(\boldsymbol{\xi}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}})$ is a sub-Gaussian random vector with variance proxy σ^2.*

The sub-Gaussian assumption is a reasonable one, and is satisfied by standard assumptions in the literature such as bounded rewards and bounded $\nabla_{\boldsymbol{\theta}} \log \pi_k(a \mid s)$ (Konda and Tsitsiklis, 2000; Papini et al., 2018). Using this assumption, we can construct an uncertainty set $\mathcal{U}_k$ that contains the true projected policy gradient with high probability.

Lemma 3.1. *Consider $\hat{\boldsymbol{\xi}}_{k,\mathbf{Q}}^{(1)}, \ldots, \hat{\boldsymbol{\xi}}_{k,\mathbf{Q}}^{(n)}$ independent, identically distributed random samples of $\boldsymbol{\xi}_{k,\mathbf{Q}}$, with $\hat{\mathbf{g}}_{k,\mathbf{Q}} = \frac{1}{n}\sum_{i=1}^n \hat{\boldsymbol{\xi}}_{k,\mathbf{Q}}^{(i)}$ their sample average. Fix $\alpha \in (0,1)$, and define*

$$\mathcal{U}_k = \left\{\mathbf{u} \mid (\mathbf{u} - \hat{\mathbf{g}}_{k,\mathbf{Q}})'\boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1}(\mathbf{u} - \hat{\mathbf{g}}_{k,\mathbf{Q}}) \leq \sigma^2 R_n^2\right\},$$

where

$$R_n^2 = \frac{1}{n}\left(\ell + 2\sqrt{\ell \log\left(\frac{1}{\alpha}\right)} + 2\log\left(\frac{1}{\alpha}\right)\right).$$

Then, $\mathbf{g}_{k,\mathbf{Q}} \in \mathcal{U}_k$ with probability at least $1-\alpha$.

Proof. The result follows by applying a concentration inequality for sub-Gaussian random vectors from Hsu et al. (2012) to a standardized version of $\hat{\mathbf{g}}_{k,\mathbf{Q}}$. See the Appendix for details. □

The ellipsoidal uncertainty set $\mathcal{U}_k$ constructed in Lemma 3.1 has an intuitive structure. It can be seen as a multivariate extension of the standard confidence interval in univariate statistics, where the radius has been calculated to accommodate the more general sub-Gaussian case (Hsu et al., 2012). Because we consider an uncertainty set centered around the projected gradient estimate, the radius only depends on the dimension ℓ of the uncertainty-aware subspace rather than the full dimension d of the policy parameterization. We now use this uncertainty set to develop a finite-sample policy improvement lower bound.

Theorem 3.1. *Consider a current policy π_k parameterized by $\boldsymbol{\theta}_k \in \mathbb{R}^d$, and any policy π parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$ where $\boldsymbol{\theta} = \boldsymbol{\theta}_k + \mathbf{Q}\mathbf{y}$ for some $\mathbf{y} \in \mathbb{R}^\ell$. Let C^{π,π_k} be defined as in Lemma 2.1, and let R_n be defined as in Lemma 3.1 with confidence parameter $\alpha \in (0,1)$. Then,*

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma}\hat{\mathbf{g}}'_{k,\mathbf{Q}}\mathbf{y} - \frac{\gamma C^{\pi,\pi_k}}{(1-\gamma)^2}\sqrt{\mathbf{y}'\mathbf{F}_{k,\mathbf{Q}}\mathbf{y}} - \frac{\sigma R_n}{1-\gamma}\sqrt{\mathbf{y}'\boldsymbol{\Sigma}_{k,\mathbf{Q}}\mathbf{y}} \tag{3.9}$$

with probability at least $1-\alpha$, up to first and second order approximation error.

Proof. Consider a robust (i.e., worst-case with respect to $\mathcal{U}_k$) lower bound of the form

$$\min_{\mathbf{u}\in\mathcal{U}_k} \frac{1}{1-\gamma}\mathbf{u}'\mathbf{y} - \frac{\gamma C^{\pi,\pi_k}}{(1-\gamma)^2}\sqrt{\mathbf{y}'\mathbf{F}_{k,\mathbf{Q}}\mathbf{y}}, \tag{3.10}$$

where $\mathcal{U}_k$ is defined as in Lemma 3.1. Note that (3.10) is the minimization of a linear function of $\mathbf{u}$ subject to a convex quadratic constraint in $\mathbf{u}$. By forming the Lagrangian and applying strong duality, we see that the minimum value of (3.10) is equivalent to the right-hand side of (3.9). By construction of $\mathcal{U}_k$, $\mathbf{g}_{k,\mathbf{Q}}$ is a feasible solution to (3.10) with probability at least $1-\alpha$. Therefore, (3.10) is a lower bound to the right-hand side of (3.8) with probability at least $1-\alpha$. This implies that (3.9) holds with probability at least $1-\alpha$, up to first and second order approximation error. See the Appendix for details. □

3.4 Uncertainty-Aware Trust Region

The appearance of an additional penalty term in our robust finite-sample lower bound from Theorem 3.1 motivates the use of a modified trust region in TRPO.

Definition 3.1 (Uncertainty-Aware Trust Region)**.** *For a given choice of trust region parameter $\delta > 0$, the* uncertainty-aware trust region *represents the set of all policies parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$ such that $\boldsymbol{\theta} = \boldsymbol{\theta}_k + \mathbf{Q}\mathbf{y}$ for some $\mathbf{y} \in \mathbb{R}^\ell$ and*

$$\frac{1}{2}\mathbf{y}'\mathbf{M}_{k,\mathbf{Q}}\mathbf{y} \leq \delta,$$

where $\mathbf{M}_{k,\mathbf{Q}} = \mathbf{F}_{k,\mathbf{Q}} + cR_n^2\boldsymbol{\Sigma}_{k,\mathbf{Q}}$ and $c \geq 0$.

Note that each term in $\mathbf{M}_{k,\mathbf{Q}}$ accounts for a main source of potential error. The first term controls the approximation error from using on-policy expectations as in TRPO, while the second term controls the finite-sample estimation error from using the projected policy gradient estimate $\hat{\mathbf{g}}_{k,\mathbf{Q}}$. The importance of including this second term is illustrated in Figure 3·2. The resulting trust region adapts to the true noise of the projected policy gradient random vector through $\boldsymbol{\Sigma}_{k,\mathbf{Q}}$, as well as the number of samples n used to estimate the projected policy gradient through the coefficient R_n^2. We include the parameter $c \geq 0$ to control the trade-off between the two terms of $\mathbf{M}_{k,\mathbf{Q}}$.

This results in a modified policy update based on the optimization problem

$$\max_{\mathbf{y}} \ \hat{\mathbf{g}}_{k,\mathbf{Q}}'\mathbf{y} \quad \text{s.t.} \quad \frac{1}{2}\mathbf{y}'\hat{\mathbf{M}}_{k,\mathbf{Q}}\mathbf{y} \leq \delta, \tag{3.11}$$

where $\hat{\mathbf{F}}_{k,\mathbf{Q}}$ in (3.7) has been replaced by a sample-based estimate $\hat{\mathbf{M}}_{k,\mathbf{Q}} = \hat{\mathbf{F}}_{k,\mathbf{Q}} + cR_n^2\hat{\boldsymbol{\Sigma}}_{k,\mathbf{Q}}$ of the uncertainty-aware trust region matrix. Finally, because (3.11) is a low-dimensional problem, we can compute its solution exactly without relying on a

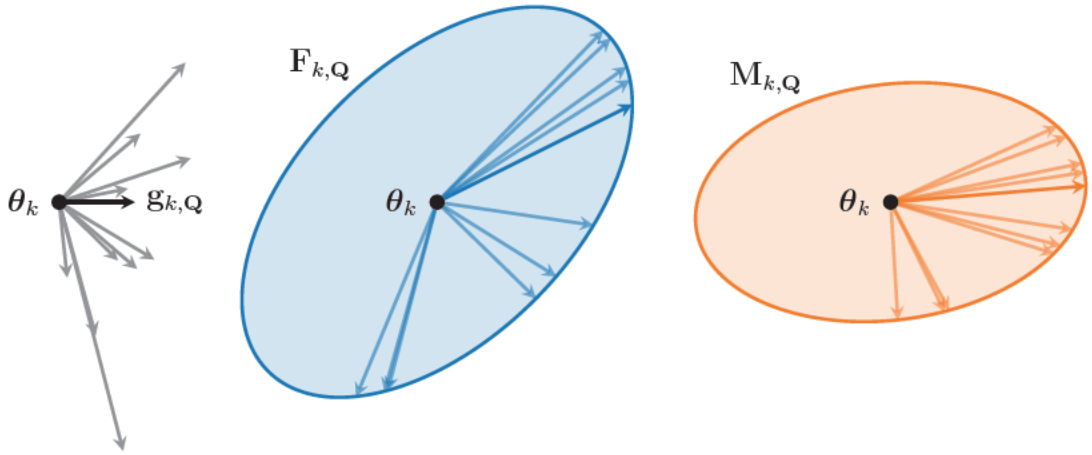

Figure 3·2: Illustration of trust regions and corresponding policy updates in parameter space for a range of sample-based projected gradient estimates. Left: True projected gradient in black, and sample-based projected gradient estimates in grey. Center: TRPO policy updates in blue. Some sample-based updates move in the opposite direction of the true projected gradient as a result of finite-sample estimation error. Right: UA-TRPO policy updates in orange. By accounting for the uncertainty present in the projected gradient estimates, all sample-based updates move in the direction of the true projected gradient.

finite number of conjugate gradient steps. The solution to (3.11) is given by

$$\mathbf{y}^* = \sqrt{\frac{2\delta}{\hat{\mathbf{g}}'_{k,\mathbf{Q}}\hat{\mathbf{M}}^{-1}_{k,\mathbf{Q}}\hat{\mathbf{g}}_{k,\mathbf{Q}}}}\,\hat{\mathbf{M}}^{-1}_{k,\mathbf{Q}}\hat{\mathbf{g}}_{k,\mathbf{Q}}. \tag{3.12}$$

3.5 Algorithm

By restricting policy updates to an uncertainty-aware subspace and applying the uncertainty-aware trust region from Definition 3.1, we develop a robust policy optimization method that adapts to the uncertainty present in the sample-based estimates of both the policy gradient and the trust region metric. These important modifications to the standard trust region approach result in our algorithm *Uncertainty-Aware Trust Region Policy Optimization (UA-TRPO)*, which is detailed in Algorithm 3.2.

Algorithm 3.2: Uncertainty-Aware Trust Region Policy Optimization

Input: initial policy π_0 with parameterization $\boldsymbol{\theta}_0 \in \mathbb{R}^d$; trust region parameters δ, c, α; random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{d \times m}$; batch size N.

for $k = 0, 1, 2, \ldots$ **do**

Collect N samples with π_k.

Use sample-based estimate of the average Fisher Information Matrix $\hat{\mathbf{F}}_k$ to construct an uncertainty-aware subspace basis $\mathbf{Q}$ via Algorithm 3.1.

Calculate projected sample-based estimates of the policy gradient $\hat{\mathbf{g}}_{k,\mathbf{Q}}$ and uncertainty-aware trust region matrix $\hat{\mathbf{M}}_{k,\mathbf{Q}} = \hat{\mathbf{F}}_{k,\mathbf{Q}} + cR_n^2\hat{\boldsymbol{\Sigma}}_{k,\mathbf{Q}}$.

Compute the solution to the uncertainty-aware policy update in (3.11) given by $\mathbf{y}^*$ in (3.12), and apply the policy update

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{Q}\mathbf{y}^*.$$

end

3.6 Experiments

In our experiments, we aim to investigate the robustness and training stability of TRPO and UA-TRPO when a limited amount of data is used for each policy update. In order to accomplish this, we perform simulations on several MuJoCo environments (Todorov et al., 2012) in OpenAI Gym (Brockman et al., 2016). In particular, we consider six continuous control locomotion tasks which vary in dimensionality: Swimmer-v3, Hopper-v3, HalfCheetah-v3, Walker2d-v3, Ant-v3, and Humanoid-v3.

Due to the complexity and high dimensionality of Ant-v3 and Humanoid-v3, we train TRPO and UA-TRPO for 10 million steps on these tasks. For all other tasks, we train for a total of 1 million steps. We run each experiment across 5 random seeds. Because we are interested in evaluating the performance of TRPO and UA-TRPO when updates must be made from limited data, we perform policy updates every $N = 1{,}024$ steps in our experiments. The tasks we consider all have a maximum time horizon of 1,000, so our choice of batch size represents as little as one full trajectory

per policy update. Most implementations of TRPO in the literature make use of larger batch sizes (Duan et al., 2016; Wu et al., 2017; Henderson et al., 2018).

With the exception of a small batch size, we consider default implementation choices commonly used in the literature for TRPO (Henderson et al., 2018; Engstrom et al., 2020; Andrychowicz et al., 2021). In particular, we represent the policy π as a multivariate Gaussian distribution, where the mean action for a given state is parameterized by a neural network with two hidden layers of 64 units each and tanh activations. The state-independent standard deviation is parameterized separately. We use a separate neural network with the same structure to parameterize our value function, and estimate advantages using Generalized Advantage Estimation (GAE) (Schulman et al., 2016). For policy updates, we consider $\epsilon = 0.2$ for the TV distance trust region radius in Definition 2.1, which leads to the KL divergence trust region radius of $\delta = 0.02$ according to Lemma 2.2.

For our implementation of UA-TRPO, we use independent standard Gaussian samples to construct the random matrix $\mathbf{\Omega}$ needed to compute random projections (Halko et al., 2011), and we consider $m = 200$ random projections to construct our uncertainty-aware subspace. We construct our uncertainty-aware trust region matrix based on 256 minibatch gradient estimates with $c = 0.1$ and $\alpha = 0.05$. See the Appendix for additional implementation details.[1]

3.6.1 Comparison with Small Batch Size

We see in Figure 3·3 that UA-TRPO generates robust policy improvement using the small batch size of $N = 1{,}024$, resulting in comparable or improved performance relative to TRPO across all environments. In addition, the benefits of UA-TRPO become more noticeable as the complexity of the task increases. In the high-dimensional Ant-v3 and Humanoid-v3 tasks, TRPO struggles to make meaningful progress over 10 mil-

[1]Code is publicly available at `https://github.com/jqueeney/uatrpo`.

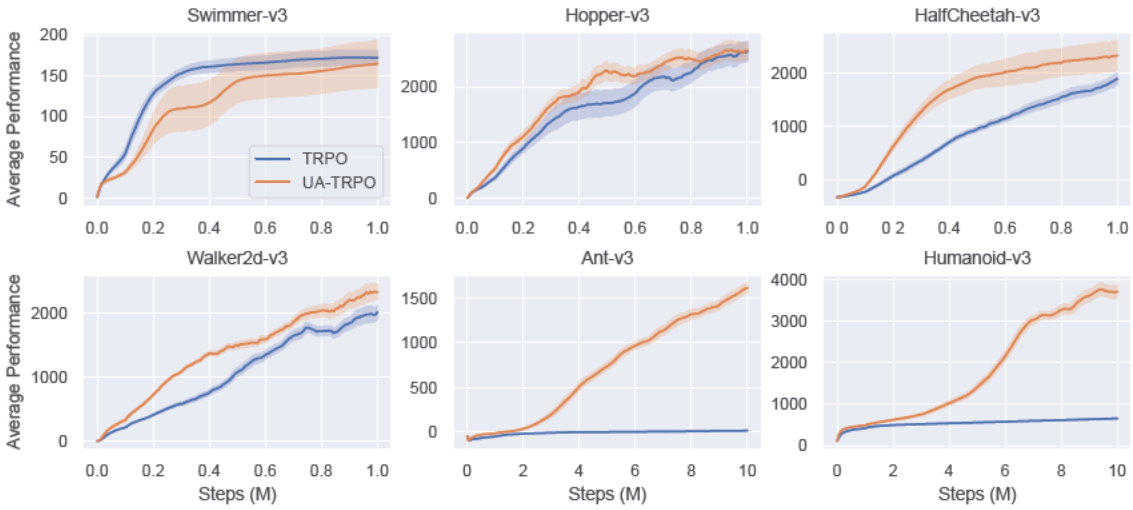

Figure 3·3: Average performance of TRPO and UA-TRPO throughout training. Shading denotes half of one standard error. Sorted in reading order from lowest to highest dimensionality.

lion steps of training when small batch sizes are used for policy updates. UA-TRPO, on the other hand, achieves consistent policy improvement throughout training and strong final performance on these difficult tasks. UA-TRPO leads to 27.1x and 6.6x the policy improvement of TRPO in Ant-v3 and Humanoid-v3, respectively. These results clearly demonstrate the advantages of our uncertainty-aware approach when small batch sizes are required for training.

3.6.2 Comparison with Adversarial Gradient Noise

We further evaluate the robustness of TRPO and UA-TRPO by introducing adversarial noise to the sample-based gradient estimates used to determine policy updates. For each dimension of the gradient, we add noise in the opposite direction of the sample-based estimate. We set the magnitude of this noise to be a multiple of standard error in every dimension of the policy gradient estimate, which allows our adversarial noise to represent plausible levels of finite-sample estimation error. In addition to the standard training case with no noise, we apply adversarial gradient noise with magnitude

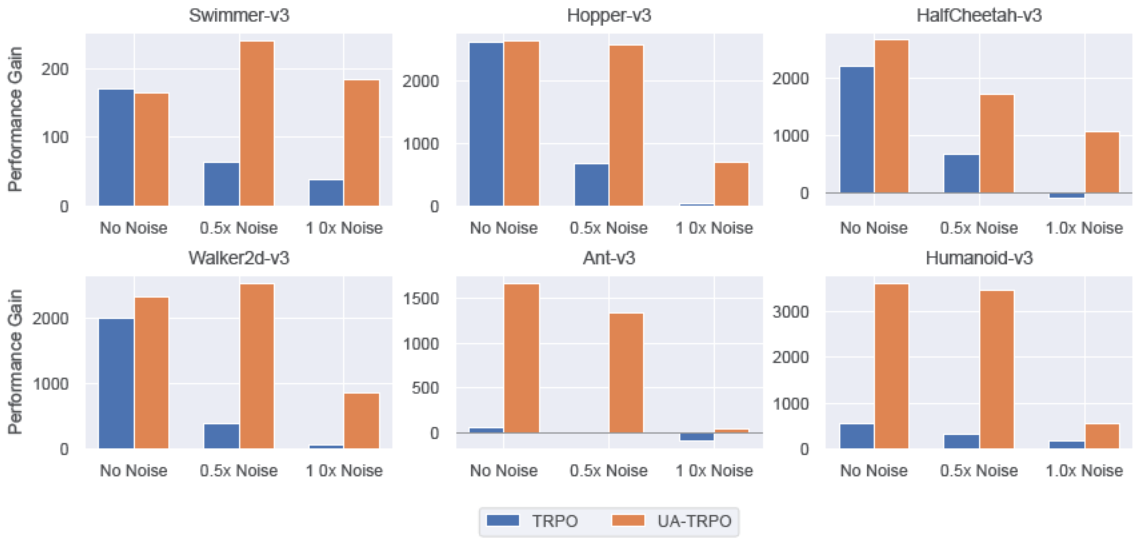

Figure 3·4: Average performance gain of TRPO and UA-TRPO from training with different levels of adversarial gradient noise. Magnitude of adversarial noise is calculated as a multiple of standard error in every dimension of the policy gradient. Performance gain evaluated after 10 million steps in Ant-v3 and Humanoid-v3, and after 1 million steps in all other tasks. Sorted in reading order from lowest to highest dimensionality.

equal to 0.5 and 1.0 times the standard error in every dimension. See Figure 3·4 for a summary of the final performance gain across tasks for each of these settings, and see the Appendix for detailed results throughout training.

The results in Figure 3·4 clearly display the robustness benefits of our uncertainty-aware modifications. For both levels of adversarial gradient noise, UA-TRPO achieves improved performance compared to TRPO in all environments. The training process of TRPO is sensitive to this adversarial noise, resulting in final performance that is worse than the initial policy in some cases. UA-TRPO, on the other hand, demonstrates consistent robustness to adversarial noise, leading to policy improvement in all experiments. In the less adversarial setting (0.5x standard error), UA-TRPO incurs minimal performance decline across tasks compared to standard training. In 4 out of 6 tasks, UA-TRPO trained in the presence of adversarial noise even surpasses the

final performance of TRPO trained without noise.

3.7 Summary

In this chapter, we have addressed the issue of sample-based uncertainty in policy improvement methods. We developed the algorithm UA-TRPO that controls the finite-sample estimation error in both the surrogate objective and trust region estimates used by TRPO. UA-TRPO is theoretically supported by a finite-sample policy improvement lower bound, which provides approximate policy improvement guarantees in the limited data setting. As a result, our algorithm demonstrates stable training from limited data and produces policy updates that are robust to noise. These represent important characteristics for reliable deep RL in real-world settings where data collection is difficult.

Because UA-TRPO is based on uncertainty-aware modifications to TRPO, it is still an on-policy algorithm that only leverages data collected under the current policy during training. In the next chapter, we develop techniques to relax the on-policy limitation of policy improvement methods, leading to improved data efficiency through sample reuse.

Chapter 4

Generalized Policy Improvement Algorithms with Sample Reuse

When data collection is expensive or limited, it is important to leverage the available samples as efficiently as possible. A natural way to accomplish this is to reuse samples to compute multiple policy updates, as done in off-policy algorithms. Unfortunately, the reuse of data from prior policies invalidates the standard performance guarantees of policy improvement algorithms, which require that samples be generated from the current policy. In this chapter, we address the on-policy limitation of existing policy improvement methods. We improve the data efficiency of on-policy algorithms through sample reuse, without sacrificing their approximate policy improvement guarantees.

4.1 Generalized Policy Improvement Lower Bound

The need for on-policy samples in existing policy improvement algorithms is a direct result of the expectations that appear in the lower bound of Lemma 2.1, which serves as the theoretical motivation for these algorithms. In order to relax the on-policy requirement of the expectations that appear in Lemma 2.1, our key insight is that we can construct a similar policy improvement bound with expectations that depend on any reference policy.

Lemma 4.1. *Consider any policy π and a reference policy π_{ref}. Then,*

$$J(\pi)-J(\pi_k) \geq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{\text{ref}}}} \left[\frac{\pi(a \mid s)}{\pi_{\text{ref}}(a \mid s)} A^{\pi_k}(s,a)\right] - \frac{2\gamma C^{\pi,\pi_k}}{(1-\gamma)^2} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\text{TV}\left(\pi, \pi_{\text{ref}}\right)(s)\right],$$

where C^{π,π_k} is defined as in Lemma 2.1.

Proof. We apply similar techniques used in Achiam et al. (2017) to prove Lemma 2.1, where the on-policy state visitation distribution d^{π_k} was the sampling distribution of interest. The key difference is that we instead consider $d^{\pi_{\text{ref}}}$ as the sampling distribution. See the Appendix for details. □

By considering our prior policies as reference policies, we can use Lemma 4.1 to develop a *Generalized Policy Improvement (GPI)* lower bound that is compatible with sample reuse.

Theorem 4.1. *Consider any policy π and prior policies π_{k-i}, $i = 0, 1, 2, \ldots$. Let ν be any choice of mixture distribution over prior policies, where $0 \leq \nu_i \leq 1$ is the probability of using π_{k-i} as the reference policy, $\sum_i \nu_i = 1$, and $\mathbb{E}_{i\sim\nu}[\,\cdot\,]$ represents an expectation determined by this mixture distribution. Then, we have that*

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{i\sim\nu} \left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}} \left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] - \frac{2\gamma C^{\pi,\pi_k}}{(1-\gamma)^2} \mathop{\mathbb{E}}_{i\sim\nu} \left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}} \left[\text{TV}\left(\pi, \pi_{k-i}\right)(s)\right]\right], \quad (4.1)$$

where C^{π,π_k} is defined as in Lemma 2.1.

Proof. For any prior policy π_{k-i}, we can apply Lemma 4.1 to construct a policy improvement lower bound with expectations that depend on the visitation distribution $d^{\pi_{k-i}}$. These all represent lower bounds on the same quantity $J(\pi) - J(\pi_k)$, so any convex combination of these lower bounds will also be a lower bound on $J(\pi) - J(\pi_k)$. Therefore, (4.1) holds for any choice of mixture distribution ν over prior policies. □

The expectations that appear in our Generalized Policy Improvement lower bound can be estimated using a mixture of samples collected under prior policies, so Theorem 4.1 provides insight into how we can reuse samples while still providing guarantees

on performance throughout training. The cost of sample reuse is that the penalty term now depends on the expected TV distance between the new policy and our prior policies, rather than the current policy. Otherwise, our lower bound remains largely unchanged compared to the on-policy case. In fact, we recover the on-policy lower bound when ν is chosen to place all weight on the current policy, so Lemma 2.1 is a special case of Theorem 4.1.

Because the structure of our generalized lower bound remains the same as the on-policy lower bound, we can use the same techniques to motivate practical policy improvement algorithms. Just as the on-policy lower bound motivated the policy update in Definition 2.1, we can use Theorem 4.1 to motivate a generalized policy update of the form

$$\begin{aligned} \max_{\pi} \; & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] \\ \text{s.t.} \; & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k-i}\right)(s)\right]\right] \leq \frac{\epsilon}{2}, \end{aligned} \tag{4.2}$$

where ϵ represents the same trust region parameter used in the on-policy case. By the triangle inequality of TV distance, we have that

$$\begin{aligned} &\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k-i}\right)(s)\right]\right] \\ &\qquad \leq \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k}\right)(s)\right]\right] + \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi_k, \pi_{k-i}\right)(s)\right]\right], \end{aligned} \tag{4.3}$$

where the first term on the right-hand side is the expected one-step TV distance and the second term does not depend on π. Therefore, we can satisfy the trust region in (4.2) by controlling the expected one-step TV distance, which is often easier to work with in practice. For an appropriate choice of ϵ_{GPI}, this leads to the following policy update.

Definition 4.1 (Generalized Trust Region Update)**.** *For a given choice of trust region*

parameter $\epsilon_{\text{GPI}} > 0$*, the* generalized trust region update *has the form*

$$\begin{aligned} \max_{\pi} \; & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] \\ \text{s.t.} \; & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{TV}\left(\pi, \pi_k\right)(s)\right]\right] \leq \frac{\epsilon_{\text{GPI}}}{2}. \end{aligned} \tag{4.4}$$

Similar to the on-policy trust region update in Definition 2.1, the generalized trust region update also provides approximate policy improvement guarantees due to its connection to the Generalized Policy Improvement lower bound in Theorem 4.1. In order to deliver these approximate policy improvement guarantees, the generalized update still depends on the advantage function with respect to the *current* policy π_k, which must be approximated using off-policy estimation techniques in practice.

As in the on-policy case, we can also satisfy the one-step TV distance trust region in (4.4) by instead considering related forward or reverse KL divergence trust regions.

Lemma 4.2. *Consider prior policies* π_{k-i}*,* $i = 0, 1, 2, \ldots$*, and any policy* π *that satisfies*

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{KL}\left(\pi_k \| \pi\right)(s)\right]\right] \leq \delta_{\text{GPI}}, \tag{4.5}$$

or

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{KL}\left(\pi \| \pi_k\right)(s)\right]\right] \leq \delta_{\text{GPI}}, \tag{4.6}$$

where $\delta_{\text{GPI}} = \epsilon_{\text{GPI}}^2/2$*. Then,* π *also satisfies the TV distance trust region in* (4.4)*.*

Proof. As in the proof of Lemma 2.2, we apply Pinsker's inequality followed by Jensen's inequality to the left-hand side of the trust region constraint in (4.4). By doing so, we have that

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{TV}\left(\pi, \pi_k\right)(s)\right]\right] \leq \sqrt{\frac{1}{2}\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{KL}\left(\pi \| \pi_k\right)(s)\right]\right]} \leq \sqrt{\frac{\delta_{\text{GPI}}}{2}} = \frac{\epsilon_{\text{GPI}}}{2}.$$

□

Next, we describe how to select the generalized trust region parameter ϵ_{GPI} and mixture distribution ν over prior policies in order to provide guarantees on the risk

of every policy update while optimizing key quantities of interest. Principled choices of ϵ_{GPI} and ν result in *theoretically supported sample reuse*.

4.2 Theoretically Supported Sample Reuse

4.2.1 Generalized Trust Region Parameter

Note that our Generalized Policy Improvement lower bound is valid for any mixture distribution ν over prior policies, and the magnitude of the penalty term depends on the choice of ν. Therefore, we first determine how to select ϵ_{GPI} for a given mixture distribution ν in order to provide the same performance guarantees as the on-policy setting.

From (4.3), we see that the generalized update in Definition 4.1 satisfies the trust region in (4.2) for any ϵ_{GPI} such that

$$\frac{\epsilon_{\text{GPI}}}{2} \leq \frac{\epsilon}{2} - \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{TV}\left(\pi_k, \pi_{k-i}\right)(s)\right]\right]. \tag{4.7}$$

While the adaptive choice of ϵ_{GPI} given by (4.7) will successfully control the magnitude of the penalty term in the Generalized Policy Improvement lower bound, it only indirectly provides insight into how ϵ_{GPI} depends on the choice of ν. In order to establish a more direct connection between ϵ_{GPI} and ν, we consider a slightly stronger trust region assumption.

Theorem 4.2. *Assume that the expected one-step TV distance under each state visitation distribution $d^{\pi_{k-i}}$ is bounded by $\epsilon_{\text{GPI}}/2$ at every update, where*

$$\epsilon_{\text{GPI}} = \frac{\epsilon}{\mathbb{E}_{i\sim\nu}\left[\,i+1\,\right]}.$$

Then, the magnitude of the generalized penalty term is no greater than the magnitude of the on-policy penalty term under the on-policy update in Definition 2.1.

Proof. The coefficients outside of the expectation in the on-policy and generalized penalty terms are the same, so we need to show that the trust region in (4.2) is

satisfied to prove the claim. For ease of notation, we write $\pi = \pi_{k+1}$. Using the triangle inequality for TV distance, we have that

$$\begin{aligned}\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi,\pi_{k-i}\right)(s)\right]\right] &\leq \mathop{\mathbb{E}}_{i\sim\nu}\left[\sum_{j=0}^{i}\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi_{k-j+1},\pi_{k-j}\right)(s)\right]\right] \\ &\leq \mathop{\mathbb{E}}_{i\sim\nu}\left[\frac{\epsilon_{\mathrm{GPI}}}{2}\cdot(i+1)\right] = \frac{\epsilon_{\mathrm{GPI}}}{2}\cdot\mathop{\mathbb{E}}_{i\sim\nu}\left[\,i+1\,\right] = \frac{\epsilon}{2},\end{aligned}$$

where we have used the assumption on the expected one-step TV distance under each state visitation distribution to bound each term inside the summation by $\epsilon_{\mathrm{GPI}}/2$. □

Although the generalized update in Definition 4.1 does not directly imply the trust region assumption in Theorem 4.2, practical implementations based on clipping mechanisms or backtracking line searches can ensure that this assumption holds. In practice, the choice of ϵ_{GPI} determined by Theorem 4.2 successfully controls the magnitude of the generalized penalty term, and tends to be conservative compared to (4.7) because it is based on applying the triangle inequality between every prior policy.

The form of ϵ_{GPI} in Theorem 4.2 clearly demonstrates the trade-off of sample reuse. As we reuse older data, we must consider smaller one-step trust regions at every policy update in order to guarantee the same level of risk.

4.2.2 Mixture Distribution

Despite the need for smaller one-step trust regions at every policy update, we can show that our generalized policy update improves key quantities of interest when the mixture distribution ν is chosen in a principled manner. Note that $\nu_i > 0$ indicates that data from π_{k-i} will be used during updates, so the choice of ν determines how many prior policies to consider in addition to how to weight their contributions.

In this section, we show it is possible to select ν in a way that improves both the effective sample size and total TV distance update size throughout training com-

pared to the on-policy baseline. It is common to consider auxiliary metrics to inform sampling schemes (Schaul et al., 2016; de Bruin et al., 2018), and these represent two important quantities in policy optimization. A larger effective sample size results in more accurate estimates of the expectations that we must approximate in policy updates, and leads to a more diverse batch of data which can be useful when reward signals are sparse (Hong et al., 2018). A larger total TV distance update size allows for more aggressive exploitation of the available information, which can lead to faster learning throughout training. Although optimizing these metrics does not guarantee improved performance compared to on-policy algorithms, we see in our experiments that empirically this is often the case.

We write the on-policy sample size as $N = Bn$, where B is a positive integer and n represents the smallest possible batch size (e.g., the length of one full trajectory). On-policy policy improvement algorithms update the policy according to Definition 2.1 after every N samples collected, where the expectations are approximated using empirical averages calculated using these N samples.

For the generalized case, we can combine data across several prior policies to construct the batch used to calculate the generalized policy update in Definition 4.1. Therefore, sample reuse allows us to make policy updates after every n samples collected. The resulting effective sample size used for generalized policy updates is given by

$$f_{\text{ESS}}(\nu) = \frac{n}{\sum_i \nu_i^2},$$

where ν_i, $i = 0, 1, 2, \ldots$, represents the weighting for data collected under π_{k-i}. We consider the effective sample size to account for the increased variance due to non-uniform weights (Kong, 1992). For the case of uniform weights over the last M policies, $f_{\text{ESS}}(\nu)$ reduces to the standard sample size definition of Mn.

Because we calculate generalized policy updates after every n samples collected,

we are able to make B times as many updates as the on-policy case. The total TV distance update size for every N samples collected in the generalized case is given by

$$f_{\mathrm{TV}}(\nu) = B \cdot \frac{\epsilon_{\mathrm{GPI}}}{2} = \frac{B}{\sum_i \nu_i \, (i+1)} \cdot \frac{\epsilon}{2},$$

compared to $\epsilon/2$ in on-policy algorithms.

Using the following result, we select ν to optimize $f_{\mathrm{ESS}}(\nu)$ and $f_{\mathrm{TV}}(\nu)$ relative to the on-policy case.

Theorem 4.3. *Fix the trade-off parameter $\kappa \in [0,1]$, and select the mixture distribution ν according to the convex optimization problem*

$$\begin{aligned} \nu^*(\kappa) = \arg\min_{\nu} \quad & \kappa \cdot \frac{\sum_i \nu_i^2}{c_{\mathrm{ESS}}} + (1-\kappa) \cdot \frac{\sum_i \nu_i \, (i+1)}{c_{\mathrm{TV}}} \\ \text{s.t.} \quad & \sum_i \nu_i^2 \leq \frac{1}{B}, \quad \sum_i \nu_i \, (i+1) \leq B, \\ & \sum_i \nu_i = 1, \quad \nu_i \geq 0, \, i = 0, 1, 2, \ldots, \end{aligned} \tag{4.8}$$

where $c_{\mathrm{ESS}}, c_{\mathrm{TV}} \geq 0$ are scaling coefficients. Then, by applying the generalized update in Definition 4.1 throughout training with ϵ_{GPI} from Theorem 4.2, the effective sample size and total TV distance update size are at least as large as the corresponding quantities in the on-policy baseline.

Proof. Note that $f_{\mathrm{ESS}}(\nu)$ and $f_{\mathrm{TV}}(\nu)$ only depend on ν in their denominators. Therefore, we can maximize these quantities by minimizing the denominators that depend on ν. By considering a convex combination determined by the trade-off parameter κ and applying scaling coefficients, we arrive at the objective in (4.8).

Next, we consider the constraints in (4.8). The first constraint implies $f_{\mathrm{ESS}}(\nu) \geq Bn$ and the second constraint implies $f_{\mathrm{TV}}(\nu) \geq \epsilon/2$. Therefore, these constraints guarantee that the effective sample size and total TV distance update size are at least as large as in the on-policy case. The remaining constraints ensure that ν is a distribution. Finally, note that a uniform distribution over the last M policies is a feasible solution to (4.8) for $B \leq M \leq 2B - 1$. Therefore, (4.8) is a feasible optimization problem. □

We see the benefits of using the optimal mixture distribution from Theorem 4.3 in

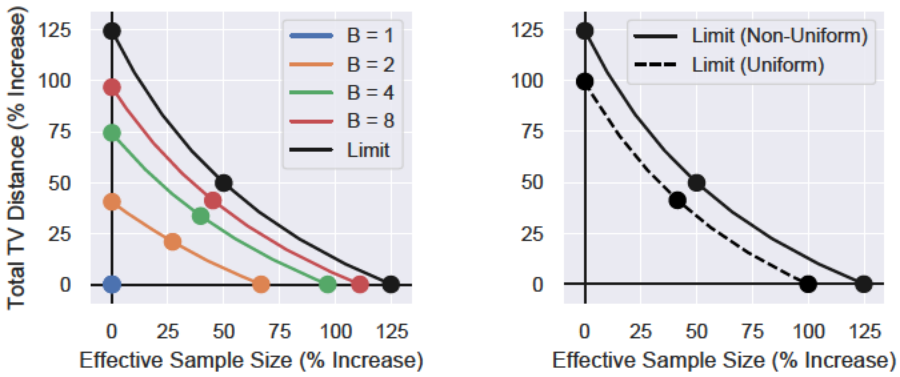

Figure 4·1: Benefit of GPI update compared to the on-policy case. Represents percent increases in effective sample size and total TV distance update size for all possible values of $\kappa \in [0, 1]$. Markers indicate $\kappa = 0.0, 0.5, 1.0$. Left: Comparison across several values of B. Right: Comparison of non-uniform and uniform weights for large B.

the left-hand side of Figure 4·1. As B becomes larger, the benefit of our generalized approach increases. A large value of B indicates the need for batches that contain many sample trajectories, which may occur in highly stochastic environments or tasks with sparse rewards. In this case, we can improve the effective sample size or total TV distance update size by up to 125% compared to on-policy methods, without sacrificing performance with respect to the other. In the right-hand side of Figure 4·1, we see that we can still achieve up to a 100% improvement in these metrics through sample reuse with uniform weights over prior policies. Therefore, the main benefit of optimizing ν comes from determining the appropriate number of prior policies to consider, with non-uniform weights offering additional improvements. Together, Theorem 4.2 and Theorem 4.3 provide theoretical support for the sample reuse we consider in this work.

In the continuous control benchmarking tasks we consider in our experiments, the default on-policy batch size used in the literature represents a low value of $B = 2$. Because these environments have deterministic dynamics, some of these tasks may require only a small number of trajectories per batch to inform policy updates. Even

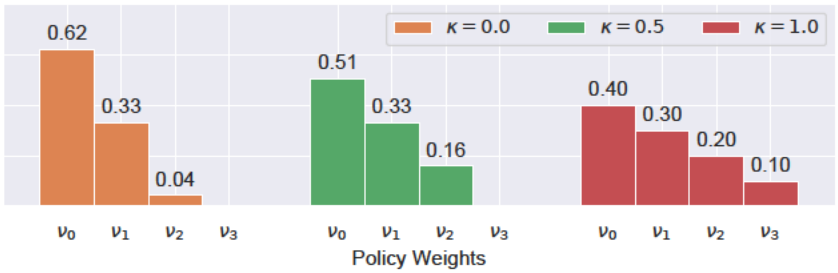

Figure 4·2: Optimal mixture distributions for $\kappa = 0.0, 0.5, 1.0$ when $B = 2$.

in this case, we can improve the effective sample size by up to 67% or total TV distance update size by up to 41% compared to on-policy methods, without sacrificing performance with respect to the other. The optimal mixture distributions for $B = 2$ are shown in Figure 4·2 for three choices of trade-off parameter κ. When $B = 2$, our theoretically supported sample reuse suggests using data from the prior three or four policies.

4.3 Algorithms

The theory that we have developed can be used to construct generalized versions of on-policy algorithms with theoretically supported sample reuse. We refer to this class of algorithms as *Generalized Policy Improvement (GPI)* algorithms. The high-level framework for these algorithms is shown in Algorithm 4.1. By using the optimal mixture distribution from Theorem 4.3, GPI algorithms only require modest increases in memory and computation compared to on-policy algorithms. In addition, due to their use of one-step trust regions, GPI algorithms only need access to the current policy and value function in order to compute updates. As a result, deep RL implementations do not require storage of any additional neural networks compared to on-policy algorithms.

In this section, we provide details on three algorithms from this class: Generalized

Algorithm 4.1: Generalized Policy Improvement Algorithms

Input: initial policy π_0; TV distance trust region parameter ϵ; on-policy batch size $N = Bn$, where n represents minimum batch size; trade-off parameter κ.

Calculate mixture distribution ν using Theorem 4.3, and let M be the number of prior policies with non-zero weighting.

Calculate generalized trust region parameter ϵ_{GPI} using Theorem 4.2.

for $k = 0, 1, 2, \ldots$ **do**

Collect n samples with π_k.

Use n samples from each of π_{k-i}, $i = 0, \ldots, M-1$, to approximate the expectations in Definition 4.1.

Update policy by approximately solving the optimization problem in Definition 4.1. Implementation varies by algorithm.

end

PPO (GePPO), Generalized TRPO (GeTRPO), and Generalized VMPO (GeVMPO).

4.3.1 Generalized PPO

PPO (Schulman et al., 2017) approximates the on-policy trust region update in Definition 2.1 using the policy update

$$\max_{\pi} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\min \left(\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a),\ \text{clip}\left(\frac{\pi(a \mid s)}{\pi_k(a \mid s)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_k}(s,a) \right) \right], \tag{4.9}$$

where $\text{clip}(x, l, u) = \min(\max(x, l), u)$ and the maximization is implemented using minibatch stochastic gradient ascent. The policy update in (4.9) approximately maximizes a lower bound on the on-policy surrogate objective, while also heuristically enforcing

$$\left| \frac{\pi(a \mid s)}{\pi_k(a \mid s)} - 1 \right| \leq \epsilon$$

at every state-action pair through the use of the clipping mechanism in the second term. The clipping mechanism accomplishes this by removing any incentive for the

probability ratio to deviate more than ϵ from its starting point during every policy update. Therefore, the clipping mechanism heuristically enforces the TV distance trust region in (2.2), as shown in the following result.

Lemma 4.3. *Assume that the support of π is contained within the support of π_k at every state. Then, the TV distance trust region in* (2.2) *can be written as*

$$\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\mathrm{TV}\left(\pi,\pi_k\right)(s)\right] = \frac{1}{2}\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}}\left[\left|\frac{\pi(a\mid s)}{\pi_k(a\mid s)}-1\right|\right].$$

Proof. From the definition of TV distance, we have that

$$\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\mathrm{TV}\left(\pi,\pi_k\right)(s)\right] = \frac{1}{2}\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\int_{\mathcal{A}}\left|\pi(a\mid s)-\pi_k(a\mid s)\right|\mathrm{d}a\right].$$

Then, by multiplying and dividing each term by $\pi_k(a \mid s)$, we obtain the result. □

Note that the assumption in Lemma 4.3 is satisfied by popular policy representations such as a Gaussian policy.

In order to develop a generalized version of PPO that approximates the generalized trust region update in Definition 4.1, we desire a policy update that approximately maximizes a lower bound on the generalized surrogate objective while also heuristically enforcing the generalized trust region via a clipping mechanism. In order to determine the appropriate clipping mechanism, we can write the generalized TV distance trust region in (4.4) as the expectation of a probability ratio deviation just as we did in the on-policy case.

Lemma 4.4. *Assume that the support of π is contained within the support of π_{k-i}, $i = 0, 1, 2, \ldots$, at every state. Then, the TV distance trust region in* (4.4) *can be written as*

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi,\pi_k\right)(s)\right]\right] = \frac{1}{2}\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\left|\frac{\pi(a\mid s)}{\pi_{k-i}(a\mid s)}-\frac{\pi_k(a\mid s)}{\pi_{k-i}(a\mid s)}\right|\right]\right].$$

Proof. Apply the same techniques as in the proof of Lemma 4.3. From the definition

of TV distance, we have that

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi,\pi_k\right)(s)\right]\right]=\frac{1}{2}\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\int_{\mathcal{A}}\left|\pi(a\mid s)-\pi_k(a\mid s)\right|\mathrm{d}a\right]\right].$$

Then, by multiplying and dividing each term by $\pi_{k-i}(a \mid s)$, we obtain the result. □

Therefore, Lemma 4.4 suggests the need for a clipping mechanism that heuristically enforces

$$\left|\frac{\pi(a\mid s)}{\pi_{k-i}(a\mid s)}-\frac{\pi_k(a\mid s)}{\pi_{k-i}(a\mid s)}\right|\leq\epsilon_{\mathrm{GPI}}$$

by removing the incentive for the probability ratio to deviate more than ϵ_{GPI} from its starting point of ${}^{\pi_k(a|s)}\!/_{\pi_{k-i}(a|s)}$. By applying such a generalized clipping mechanism and considering a lower bound on the generalized surrogate objective, we arrive at the Generalized PPO (GePPO) update

$$\begin{aligned}\max_{\pi}\ \mathop{\mathbb{E}}_{i\sim\nu}\Bigg[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\Bigg[&\min\Bigg(\frac{\pi(a\mid s)}{\pi_{k-i}(a\mid s)}A^{\pi_k}(s,a),\\&\mathrm{clip}\left(\frac{\pi(a\mid s)}{\pi_{k-i}(a\mid s)},\ \frac{\pi_k(a\mid s)}{\pi_{k-i}(a\mid s)}-\epsilon_{\mathrm{GPI}},\ \frac{\pi_k(a\mid s)}{\pi_{k-i}(a\mid s)}+\epsilon_{\mathrm{GPI}}\right)A^{\pi_k}(s,a)\Bigg)\Bigg]\Bigg].\end{aligned}\tag{4.10}$$

Because the GePPO objective in (4.10) is maximized using minibatch stochastic gradient ascent, the effectiveness of the clipping mechanism in enforcing the corresponding trust region depends on the learning rate. To see why this is true, note that each probability ratio begins at the center of the clipping range at the start of each policy update. Therefore, the clipping mechanism has no impact at the beginning of each update, and a large learning rate can result in probability ratios far outside of the clipping range (Engstrom et al., 2020). In order to address this concern, we propose an adaptive learning rate where we decrease the learning rate if the expected TV distance of a policy update exceeds the target trust region radius. See Algorithm 4.2 for details. This allows the theoretical connection between the clipping mechanism

Algorithm 4.2: GePPO Adaptive Learning Rate

Input: TV distance trust region parameter ϵ_{GPI}; policy learning rate η; adaptive learning rate factor $\upsilon \geq 0$.

Calculate sample-based estimate $\widehat{\text{TV}}$ of one-step TV distance trust region in Definition 4.1 using Lemma 4.4.

if $\widehat{\text{TV}} > \epsilon_{\text{GPI}}/2$ **then** $\eta = \eta/(1+\upsilon)$.

and the TV distance trust region to also hold in practice. Note that this is similar to decaying learning rate schedules which are commonly used in PPO (Engstrom et al., 2020; Andrychowicz et al., 2021), but our approach automatically adapts to satisfy the goal of approximate policy improvement. In our experiments, we apply this adaptive learning rate to both PPO and GePPO.

4.3.2 Generalized TRPO

As described in Chapter 3, TRPO (Schulman et al., 2015) approximates the on-policy update in Definition 2.1 by instead applying the forward KL divergence trust region in (2.3). Therefore, TRPO considers the policy update

$$\max_{\pi} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a) \right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\text{KL}\left(\pi_k \| \pi\right)(s) \right] \leq \delta.$$

Next, TRPO considers a first order expansion of the surrogate objective and second order expansion of the forward KL divergence trust region with respect to the policy parameterization. Using these approximations, the TRPO policy update admits a closed-form solution for the parameterization of π_{k+1}. Finally, a backtracking line search is performed to guarantee that the trust region is satisfied. See Chapter 3 for additional details.

It is straightforward to extend TRPO to the generalized setting. We approximate the GPI update in Definition 4.1 by instead applying the forward KL divergence trust

region in (4.5). This leads to the policy update

$$\begin{aligned} \max_{\pi} \; & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] \\ \text{s.t.} \; & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{KL}\left(\pi_k \| \pi\right)(s)\right]\right] \leq \delta_{\mathrm{GPI}}. \end{aligned} \tag{4.11}$$

We consider the same approximations and optimization procedure as TRPO to implement the Generalized TRPO (GeTRPO) update. For policies π and π_k parameterized by $\boldsymbol{\theta}, \boldsymbol{\theta}_k \in \mathbb{R}^d$, respectively, approximations to the generalized surrogate objective and trust region in (4.11) are given by

$$\begin{aligned} \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] &\approx \mathbf{g}'_{k,\nu}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_k\right), \\ \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{KL}\left(\pi_k \| \pi\right)(s)\right]\right] &\approx \frac{1}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_k\right)' \mathbf{F}_{k,\nu}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_k\right), \end{aligned}$$

which represent first and second order expansions, respectively, centered around π_k.

4.3.3 Generalized VMPO

VMPO (Song et al., 2020) approximates the on-policy update in Definition 2.1 by instead considering the reverse KL divergence trust region in (2.4). VMPO begins by calculating a non-parametric target policy based on this update. However, in the on-policy setting we can only compute advantages for state-action pairs we have visited, so VMPO first transforms the update to treat the state-action visitation distribution as the variable rather than the policy. We write the new and current state-action visitation distributions ψ, ψ_k as

$$\psi(s,a) = d^{\pi_k}(s)\pi(a \mid s), \qquad \psi_k(s,a) = d^{\pi_k}(s)\pi_k(a \mid s),$$

which results in the non-parametric VMPO update

$$\psi_{\text{targ}} = \arg\max_{\psi} \mathop{\mathbb{E}}_{(s,a)\sim\psi} \left[A^{\pi_k}(s,a)\right] \quad \text{s.t.} \quad \text{KL}\left(\psi \| \psi_k\right) \leq \delta. \tag{4.12}$$

This leads to the target distribution

$$\psi_{\text{targ}}(s,a) = d^{\pi_k}(s)\pi_k(a \mid s)w(s,a),$$

where

$$w(s,a) = \frac{\exp\left(A^{\pi_k}(s,a)/\lambda^*\right)}{Z(\lambda^*)}, \qquad Z(\lambda^*) = \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\exp\left(A^{\pi_k}(s,a)/\lambda^*\right)\right],$$

and

$$\lambda^* = \arg\min_{\lambda \geq 0} \ \lambda\delta + \lambda \log\left(Z(\lambda)\right)$$

is the optimal solution to the corresponding dual problem.

Next, VMPO projects this target distribution back onto the space of parametric policies, while guaranteeing that the new policy satisfies the forward KL divergence trust region in (2.3). Therefore, VMPO guarantees approximate policy improvement in both the initial non-parametric step and the subsequent projection step. This results in the constrained maximum likelihood update

$$\max_{\pi} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[w(s,a) \log \pi(a \mid s)\right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\text{KL}\left(\pi_k \| \pi\right)(s)\right] \leq \delta. \tag{4.13}$$

In order to generalize VMPO, we begin by approximating the GPI update in Definition 4.1 with the reverse KL divergence trust region from (4.6). In the generalized setting, the new and current state-action visitation distributions ψ, ψ_k used in the non-parametric update step are given by

$$\psi(s,a) = \mathop{\mathbb{E}}_{i\sim\nu} \left[d^{\pi_{k-i}}(s)\right] \pi(a \mid s), \qquad \psi_k(s,a) = \mathop{\mathbb{E}}_{i\sim\nu} \left[d^{\pi_{k-i}}(s)\right] \pi_k(a \mid s).$$

Using these generalized visitation distributions, the non-parametric update has the same form as (4.12) with δ replaced by δ_{GPI}. This results in the target distribution

$$\psi_{\text{targ}}(s,a) = \mathop{\mathbb{E}}_{i\sim\nu}\left[d^{\pi_{k-i}}(s)\right]\pi_k(a \mid s)w(s,a),$$

where $w(s,a)$ has the same form as in the on-policy case, the normalizing coefficient is now given by

$$Z(\lambda^*) = \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)}\exp\left(A^{\pi_k}(s,a)/\lambda^*\right)\right]\right],$$

and λ^* is the optimal solution to the corresponding dual problem as in the on-policy case.

The projection step in the generalized case considers the forward KL divergence trust region in (4.5), resulting in the Generalized VMPO (GeVMPO) update

$$\begin{aligned}\max_{\pi}\ & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)}w(s,a)\log\pi(a \mid s)\right]\right]\\ \text{s.t.}\ & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{KL}\left(\pi_k\|\pi\right)(s)\right]\right] \leq \delta_{\text{GPI}}.\end{aligned} \tag{4.14}$$

In order to implement the GeVMPO update in (4.14), we approximate expectations using sample averages. Because we only have access to a single action in any given state, the empirical version of the maximum likelihood objective in (4.14) incentivizes an increase in likelihood at every state-action pair, even those with $w(s,a) < 1$. In order to address this issue, Song et al. (2020) only considered samples with positive advantages. Rather than discard potentially useful information from half of the collected samples, we propose an alternative approach.

We note the similarity between the GeVMPO update in (4.14) and the GeTRPO update in (4.11), where the only difference comes in the form of the objective. Therefore, we can apply the same optimization procedure as in GeTRPO, where we consider

a first order approximation of the objective and a second order approximation of the forward KL divergence trust region. Similar to policy gradient methods, we introduce a baseline value to the objective that does not impact the gradient at the current policy π_k (Sutton et al., 2000). In particular, we replace the non-parametric target weights $w(s, a)$ in (4.14) with $\bar{w}(s, a) = w(s, a) - 1$. This leads to the same gradient as the true maximum likelihood objective at π_k, but results in the appropriate update direction at every state-action pair when we approximate expectations using sample averages. In our experiments, we apply these implementation details to both VMPO and GeVMPO.

4.4 Experiments

In order to analyze the performance of our GPI algorithms, we consider the full set of 28 continuous control benchmark tasks in the DeepMind Control Suite (Tunyasuvunakool et al., 2020). This benchmark set covers a broad range of continuous control tasks, including a variety of classic control, goal-oriented manipulation, and locomotion tasks. In addition, the benchmark tasks vary in complexity, both in terms of dimensionality and sparsity of reward signals. Finally, each task has a horizon length of 1,000 and $r(s, a) \in [0, 1]$ for every state-action pair, resulting in a total return between 0 and 1,000.

We focus our analysis on the comparison between GPI algorithms and their on-policy policy improvement counterparts. Note that we do not claim state-of-the-art performance, but instead we are interested in evaluating the benefits of theoretically supported sample reuse in the context of policy improvement algorithms. By doing so, we can support the use of GPI algorithms in settings where on-policy methods are currently the most viable option for data-driven control.

In our experiments, we consider default network architectures and hyperparam-

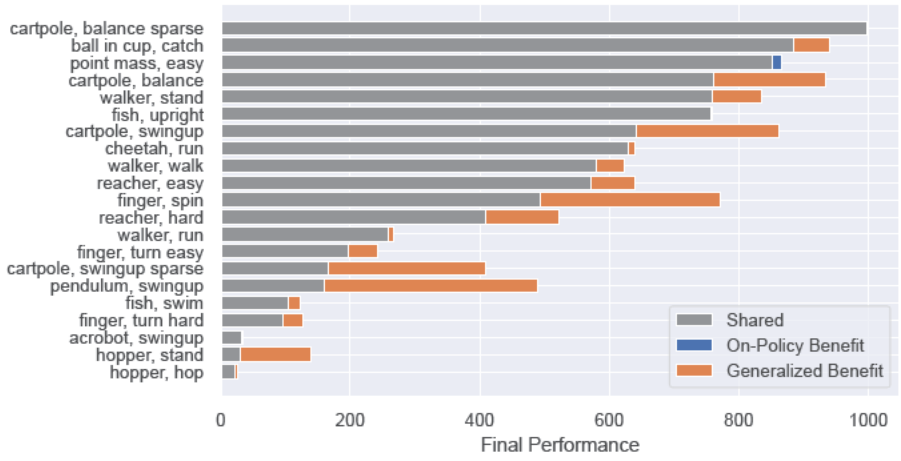

Figure 4·3: Generalized vs. on-policy final performance by task. Bars represent final performance of the best performing GPI algorithm and the best performing on-policy algorithm. Excludes 7 tasks where no learning occurs under any algorithm. Sorted from high to low based on on-policy performance.

eters commonly found in the literature (Henderson et al., 2018; Engstrom et al., 2020; Andrychowicz et al., 2021). We model the policy π as a multivariate Gaussian distribution with diagonal covariance, where the mean action for a given state is parameterized by a neural network with two hidden layers of 64 units each and tanh activations. The state-independent standard deviation of each action dimension is parameterized separately. We represent the value function using a neural network with the same structure. For each task, we train our policy for a total of 1 million steps, and we average over 5 random seeds. Note that some of the more difficult, high-dimensional tasks do not demonstrate meaningful learning under these default choices for any of the policy improvement algorithms we consider. These tasks likely require significantly longer training, different network architectures, or other algorithmic frameworks to successfully learn.

We evaluate performance across three on-policy policy improvement algorithms: PPO, TRPO, and VMPO. For these on-policy algorithms, we consider the default

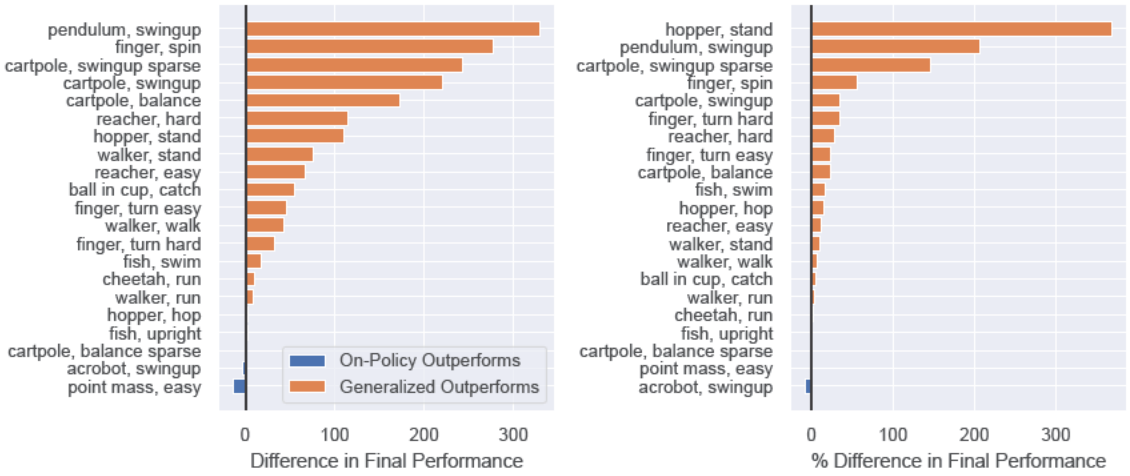

Figure 4·4: Difference between generalized and on-policy final performance by task. Bars represent difference in final performance between the best performing GPI algorithm and the best performing on-policy algorithm. Excludes 7 tasks where no learning occurs under any algorithm. Sorted from high to low. Left: Total difference. Right: Percent difference.

batch size of $N = 2{,}048$, which we write as $B = 2$ and $n = 1{,}024$ since every task we consider has a horizon length of 1,000. We consider $\epsilon = 0.2$ for the TV distance trust region parameter in Definition 2.1, which corresponds to the clipping parameter ϵ in PPO. We calculate the KL divergence trust region parameter for TRPO and VMPO according to Lemma 2.2, which results in $\delta = 0.02$. We estimate advantages using Generalized Advantage Estimation (GAE) (Schulman et al., 2016).

We also train policies using generalized versions of each on-policy algorithm: GePPO, GeTRPO, and GeVMPO. When selecting the mixture distribution over prior policies according to Theorem 4.3, we consider the trade-off parameter with the best final performance from the set of $\kappa = 0.0, 0.5, 1.0$. These choices of trade-off parameter lead to the policy weights in Figure 4·2. The generalized trust region parameters ϵ_{GPI} and δ_{GPI} are calculated according to Theorem 4.2 and Lemma 4.2, respectively. As in the on-policy case, our GPI algorithms require estimates of $A^{\pi_k}(s, a)$. In order to

Table 4.1: Task classification by algorithm.

Task Classification	PPO	TRPO	VMPO	Best
On-Policy Outperforms	3	1	3	2
Generalized Outperforms	18	19	17	19
No Learning	7	8	8	7
Total Number of Tasks	28	28	28	28

accomplish this, we consider an off-policy variant of GAE that uses the V-trace value function estimator (Espeholt et al., 2018). See the Appendix for details, including the values of all hyperparameters.[1]

4.4.1 Overview of Experimental Results

In order to evaluate the benefit of our generalized framework, we compare the best performing on-policy algorithm to the best performing GPI algorithm for every task where learning occurs. Figure 4·3 shows the final performance of these algorithms by task, and Figure 4·4 shows the difference in final performance by task. From these results, we see a clear performance gain from our generalized approach. Out of 28 tasks, our GPI algorithms outperform in 19 tasks, on-policy algorithms outperform in 2 tasks, and we observe no meaningful learning under any algorithm in 7 tasks when compared to a random policy. In the 2 tasks where on-policy algorithms outperform, the performance difference is small. On the other hand, in the tasks where our GPI algorithms outperform, we often observe a significant performance difference. We see an improvement of more than 50% from our GPI algorithms in 4 of the tasks and an improvement of more than 10% in 13 of the tasks. Note that we also observe similar trends when comparing any on-policy algorithm to its corresponding generalized version, as summarized in Table 4.1. See the Appendix for details.

[1]Code is publicly available at `https://github.com/jqueeney/gpi`.

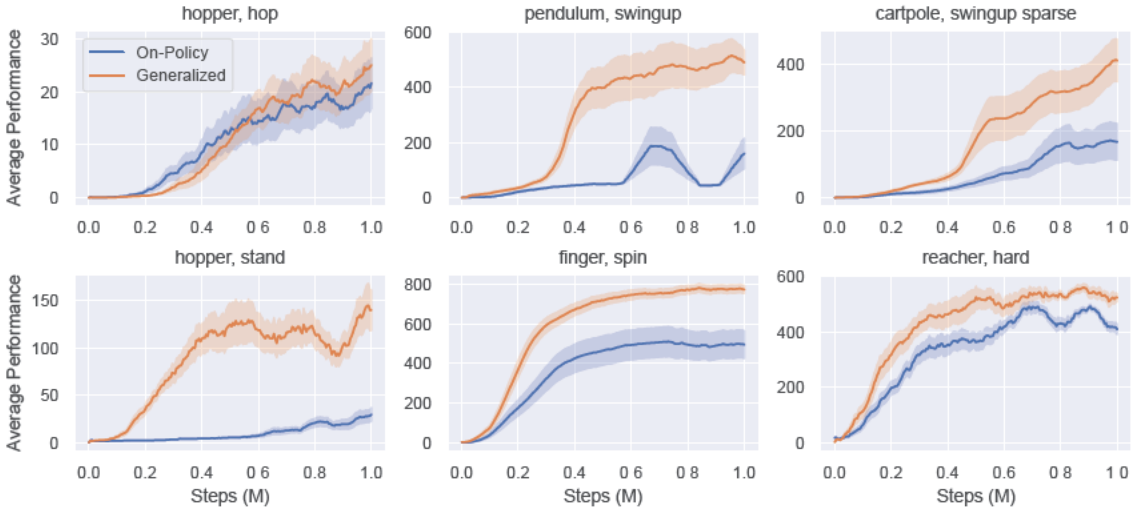

Figure 4·5: Generalized vs. on-policy performance throughout training for sparse reward tasks. Training curves represent performance of the best performing GPI algorithm and the best performing on-policy algorithm. Shading denotes half of one standard error. Sorted in reading order from most to least sparse.

4.4.2 Analysis of Sparse Reward Tasks

From Figure 4·4, it is clear that our generalized framework results in improved performance across a broad range of tasks. This benefit is most pronounced for tasks with sparse reward signals that are difficult to find and exploit. In order to quantify the sparsity of every task in the benchmarking set, we measure the percentage of samples that contain a non-negligible reward signal under a random policy. We measure this statistic by collecting 100,000 samples under a random policy, and calculating the percentage of these samples where $r(s, a) > 0.01$.

In 6 of the benchmarking tasks where learning occurs, a random policy receives a reward signal less than 1% of the time. These represent the tasks with the highest level of sparsity, and Figure 4·5 shows the performance throughout training for each of them. We note that our generalized approach results in improved performance in all of these sparse reward tasks. In fact, the 3 tasks where our GPI algorithms demonstrate

the largest total gain in performance and the 4 tasks where our GPI algorithms demonstrate the largest percentage gain in performance are all contained in this set of sparse reward tasks. In this setting, on-policy algorithms struggle to exploit the limited reward information. The sample reuse in our GPI algorithms, on the other hand, allows sparse reward signals to be exploited across several policy updates while also leading to larger, more diverse batches of data at every update. Together, these benefits of sample reuse result in improved learning progress in difficult sparse reward settings.

4.5 Summary

In this chapter, we have addressed the on-policy limitation of existing policy improvement methods. We developed a class of Generalized Policy Improvement algorithms that guarantee approximate policy improvement throughout training while reusing data from all recent policies. We demonstrated the theoretical benefits of principled sample reuse, and showed empirically that our generalized approach results in improved performance compared to popular on-policy algorithms. Therefore, our class of GPI algorithms represents a strong alternative in settings where on-policy methods are currently the default choice for data-driven decision making and control, providing the same guarantees on stable training with improved data efficiency.

Because our methods use on-policy policy improvement algorithms as a starting point, our Generalized Policy Improvement lower bound only supports the reuse of data from recent policies. For settings where large replay buffers are feasible, an interesting avenue for future work includes the development of policy improvement guarantees that are compatible with the aggressive sample reuse in off-policy algorithms. We provide a discussion on this future research direction in Chapter 7.

Chapter 5

Optimal Transport Perturbations with Safety Constraints

We now turn our attention to the issues of robustness and safety in deep RL, which are critical for the reliable deployment of learned control policies in many important application areas. We consider deep RL with safety constraints, and we incorporate uncertainty about the true environment at deployment time (i.e., model uncertainty). In this chapter, we focus on worst-case transition models in an uncertainty set, which is a common formulation of model uncertainty in robust RL. Unfortunately, many choices of uncertainty sets can be difficult to implement in a way that is compatible with the training requirements of real-world settings. In order to address this concern, we consider an uncertainty set based on the optimal transport cost between transition models, which provides robustness to general forms of environment disturbances while only requiring standard data collection from a single training environment.

5.1 Robust and Safe Reinforcement Learning

Throughout this chapter, we consider the RC-MDP framework with uncertainty set $\mathcal{P} = \bigotimes_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{P}_{s,a}$, which leads to the robust and safe RL problem in (2.7). As in the standard safe RL setting, we can apply off-policy optimization techniques to iteratively optimize (2.7) by considering the related optimization problem

$$\max_{\pi} \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \left[Q^{\pi_k}_{\mathcal{P},r}(s,a) \right] \right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \left[Q^{\pi_k}_{\mathcal{P},c}(s,a) \right] \right] \leq B, \tag{5.1}$$

where π_k is the current policy, $\mathcal{D}$ is a replay buffer containing data collected during training, and $Q^{\pi}_{\mathcal{P},r}(s,a)$ and $Q^{\pi}_{\mathcal{P},c}(s,a)$ represent robust Q functions. Compared to the standard safe RL update in (2.6), the only difference in the robust and safe RL update of (5.1) comes from the use of robust Q functions. Therefore, in order to incorporate robustness into existing deep safe RL algorithms, we focus on how to efficiently learn the robust Q functions that are needed for the policy update in (5.1).

The robust Q functions $Q^{\pi}_{\mathcal{P},r}(s,a)$ and $Q^{\pi}_{\mathcal{P},c}(s,a)$ represent the unique fixed points of the corresponding robust Bellman operators (Nilim and Ghaoui, 2005; Iyengar, 2005) given by

$$\mathcal{T}^{\pi}_{\mathcal{P},r}Q_r(s,a) := r(s,a) + \gamma \inf_{p_{s,a}\in\mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[V^{\pi}_r(s')\right], \tag{5.2}$$

$$\mathcal{T}^{\pi}_{\mathcal{P},c}Q_c(s,a) := c(s,a) + \gamma \sup_{p_{s,a}\in\mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[V^{\pi}_c(s')\right], \tag{5.3}$$

where we write $V^{\pi}_r(s') = \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q_r(s',a')\right]$ and $V^{\pi}_c(s') = \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q_c(s',a')\right]$. Note that $\mathcal{T}^{\pi}_{\mathcal{P},r}$ and $\mathcal{T}^{\pi}_{\mathcal{P},c}$ are contraction operators (Nilim and Ghaoui, 2005; Iyengar, 2005), so we can apply standard temporal difference learning techniques to learn $Q^{\pi}_{\mathcal{P},r}(s,a)$ and $Q^{\pi}_{\mathcal{P},c}(s,a)$. In order to do so, we must be able to calculate the Bellman targets in (5.2) and (5.3), which involve optimization problems over transition models that depend on the choice of uncertainty set $\mathcal{P}_{s,a}$ at every state-action pair. In order to efficiently estimate these Bellman targets, popular choices of $\mathcal{P}_{s,a}$ in the literature require the ability to change physical parameters of the environment (Mankowitz et al., 2020, 2021) or directly apply adversarial perturbations during trajectory roll-outs (Pinto et al., 2017; Tessler et al., 2019a; Vinitsky et al., 2020) to calculate worst-case transitions. However, because these implementations rely on multiple simulated training environments or potentially dangerous adversarial interventions, they are not compatible with settings that require real-world data collection for training.

5.2 Optimal Transport Uncertainty Set

In this work, we use optimal transport theory to consider an uncertainty set that can be efficiently implemented in a model-free fashion using only samples collected from a nominal training environment. In order to do so, we assume that $\mathcal{S}$ is a Polish space (i.e., a separable, completely metrizable topological space). Note that the Euclidean space $\mathbb{R}^m$ is Polish, so this is not very restrictive. Next, we define $\mathcal{P}_{s,a}$ using the optimal transport cost between transition models.

Definition 5.1 (Optimal Transport Cost)**.** *Let $\mathcal{S}$ be a Polish space, and let $d_{s,a} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_+$ be a non-negative, lower semicontinuous function satisfying $d_{s,a}(s', s') = 0$ for all $s' \in \mathcal{S}$. Then, the* optimal transport cost *between two transition models $\hat{p}_{s,a}, p_{s,a} \in P(\mathcal{S})$ is defined as*

$$\mathrm{OTC}_{d_{s,a}}(\hat{p}_{s,a}, p_{s,a}) = \inf_{\nu \in \Gamma(\hat{p}_{s,a}, p_{s,a})} \int_{\mathcal{S}\times\mathcal{S}} d_{s,a}(\hat{s}', s') \mathrm{d}\nu(\hat{s}', s'),$$

where $\Gamma(\hat{p}_{s,a}, p_{s,a})$ is the set of all couplings of $\hat{p}_{s,a}$ and $p_{s,a}$.

See Figure 5·1 for an illustration of Definition 5.1. If $d_{s,a}$ is chosen to be a metric raised to some power $p \geq 1$, we recover the p-Wasserstein distance raised to the power p as a special case (Chen and Paschalidis, 2020). If we let $d_{s,a}(\hat{s}', s') = \mathbf{1}_{\hat{s}' \neq s'}$, we recover the TV distance as a special case (Villani, 2008).

By considering the optimal transport cost from some nominal transition model $\hat{p}_{s,a}$, we define the optimal transport uncertainty set as follows.

Definition 5.2 (Optimal Transport Uncertainty Set)**.** *For a given nominal transition model $\hat{p}_{s,a}$, transport cost function $d_{s,a}$, and radius $\epsilon_{s,a}$ at $(s, a) \in \mathcal{S} \times \mathcal{A}$, the* optimal transport uncertainty set *is defined as*

$$\mathcal{P}_{s,a} = \left\{ p_{s,a} \in P(\mathcal{S}) \mid \mathrm{OTC}_{d_{s,a}}(\hat{p}_{s,a}, p_{s,a}) \leq \epsilon_{s,a} \right\}.$$

This uncertainty set has previously been considered in robust RL for the special

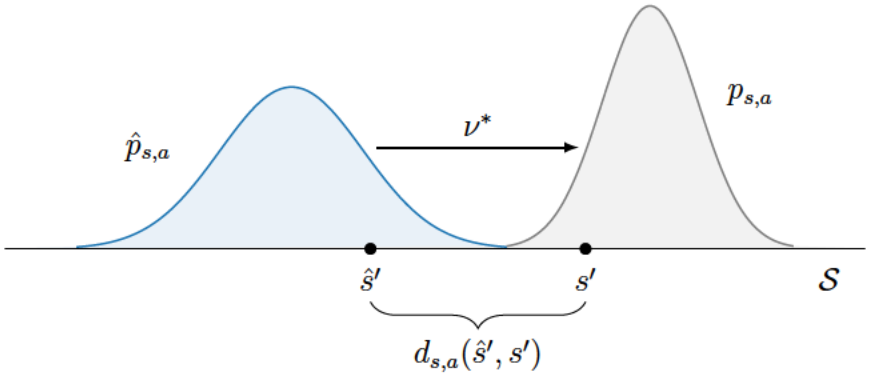

Figure 5·1: Illustration of optimal transport cost between transition models $\hat{p}_{s,a}, p_{s,a} \in P(\mathcal{S})$. Informally, $\text{OTC}_{d_{s,a}}(\hat{p}_{s,a}, p_{s,a})$ represents the cost of transporting the probability mass of $\hat{p}_{s,a}$ to $p_{s,a}$ using the optimal (i.e., minimum cost) transport plan ν^*, where the transport cost is determined by $d_{s,a}$.

case of the Wasserstein distance (Abdullah et al., 2019; Hou et al., 2020; Kuang et al., 2022).

The use of optimal transport cost to compare transition models has several benefits. First, optimal transport cost accounts for the relationship between states in $\mathcal{S}$ through the function $d_{s,a}$, and we can choose $d_{s,a}$ to reflect the geometry of $\mathcal{S}$ in a meaningful way. In particular, optimal transport cost allows significant flexibility in the choice of $d_{s,a}$, including threshold-based binary comparisons (Pydi and Jog, 2020) and percentage-based comparisons between states that are not metrics or pseudo-metrics. In our experiments where $\mathcal{S} = \mathbb{R}^m$, for example, we consider a percentage-based comparison of state transitions given by

$$d_{s,a}(\hat{s}', s') = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{s'_i - s_i}{\hat{s}'_i - s_i} - 1 \right)^2, \tag{5.4}$$

with the convention that $0/0 = 1$ so that $d_{s,a}(s', s') = 0$ for all $s' \in \mathcal{S}$. Note that (5.4) satisfies all requirements in Definition 5.1, and is not a metric or pseudo-metric.

Next, note that optimal transport cost remains valid for distributions that do not

share the same support, unlike other popular measures between distributions such as the KL divergence. As a result, the optimal transport uncertainty set is very general and can be applied to both stochastic and deterministic transition models. Finally, as we will show in the following sections, the use of an optimal transport uncertainty set results in an efficient model-free implementation of robust and safe RL *that only requires the ability to collect data in a nominal training environment.*

5.3 Reformulation as Worst-Case Virtual State Transitions

In order to learn robust Q functions for an optimal transport uncertainty set, we focus on how to efficiently calculate the robust Bellman operators in (5.2) and (5.3). We consider the following main assumptions.

Assumption 5.1. *For any* π *and* $Q_r(s', a')$ *in* (5.2), $V_r^\pi(s') = \mathbb{E}_{a' \sim \pi(\cdot|s')}\left[Q_r(s', a')\right]$ *is lower semicontinuous and* $\mathbb{E}_{s' \sim \hat{p}_{s,a}} |V_r^\pi(s')| < \infty$. *For any* π *and* $Q_c(s', a')$ *in* (5.3), $V_c^\pi(s') = \mathbb{E}_{a' \sim \pi(\cdot|s')}\left[Q_c(s', a')\right]$ *is upper semicontinuous and* $\mathbb{E}_{s' \sim \hat{p}_{s,a}} |V_c^\pi(s')| < \infty$.

Assumption 5.2. *Optimal transport plans exist for the distributionally robust optimization problems in* (5.2) *and* (5.3).

Note that Assumptions 5.1–5.2 correspond to assumptions in Blanchet and Murthy (2019) applied to our setting. In practice, the use of neural network representations results in continuous value functions, which are bounded for the common case when rewards and costs are bounded, respectively. A sufficient condition for Assumption 5.2 to hold is if $\mathcal{S}$ is compact, or if we restrict our attention to a compact subset of next states in our definition of $\mathcal{P}_{s,a}$ which is reasonable in practice. Blanchet and Murthy (2019) also provide other sufficient conditions for Assumption 5.2 to hold.

Under these assumptions, we can reformulate the Bellman operators in (5.2) and (5.3) to arrive at a tractable result that can be efficiently implemented in a deep RL setting.

Theorem 5.1. *Let Assumptions 5.1–5.2 hold, and let $\mathcal{G}$ be the set of all functions from $\mathcal{S}$ to $\mathcal{S}$. Then, we have*

$$\mathcal{T}^{\pi}_{\mathcal{P},r} Q_r(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[V^{\pi}_r(g^r_{s,a}(\hat{s}')) \right], \tag{5.5}$$

$$\mathcal{T}^{\pi}_{\mathcal{P},c} Q_c(s,a) = c(s,a) + \gamma \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[V^{\pi}_c(g^c_{s,a}(\hat{s}')) \right], \tag{5.6}$$

where for a given state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$ we have

$$g^r_{s,a} \in \arg\min_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} [V^{\pi}_r(g(\hat{s}'))] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} [d_{s,a}(\hat{s}', g(\hat{s}'))] \leq \epsilon_{s,a}, \tag{5.7}$$

$$g^c_{s,a} \in \arg\max_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} [V^{\pi}_c(g(\hat{s}'))] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} [d_{s,a}(\hat{s}', g(\hat{s}'))] \leq \epsilon_{s,a}. \tag{5.8}$$

Proof. First, we leverage results from Blanchet and Murthy (2019) to show that optimal transport strong duality holds for the distributionally robust optimization problems in (5.2) and (5.3) under Assumption 5.1. Next, we establish that the distributionally robust optimization problems in (5.2) and (5.3) share the same dual problems as (5.7) and (5.8), respectively. Finally, we use Assumption 5.2 to show that strong duality also holds for (5.7) and (5.8), which proves the result. See the Appendix for details. □

Theorem 5.1 demonstrates that we can calculate the Bellman operators $\mathcal{T}^{\pi}_{\mathcal{P},r}$ and $\mathcal{T}^{\pi}_{\mathcal{P},c}$ by using samples collected from a nominal environment with transition model $\hat{p}_{s,a}$, and adversarially perturbing the next state samples according to (5.7) and (5.8), respectively. We refer to the resulting changes in state transitions as *Optimal Transport Perturbations (OTP)*. As a result, we have replaced difficult optimization problems over distribution space in (5.2) and (5.3) with the tractable problems of computing Optimal Transport Perturbations in state space. See Figure 5·2 for an illustration. Theorem 5.1 represents the main theoretical contribution of this chapter, which directly motivates an efficient deep RL implementation of robust and safe RL.

Finally, note that these perturbed state transitions are only needed to calculate the Bellman targets in (5.5) and (5.6), which we use to train the robust Q functions $Q^{\pi}_{\mathcal{P},r}(s,a)$ and $Q^{\pi}_{\mathcal{P},c}(s,a)$. Therefore, unlike other adversarial approaches to robust RL

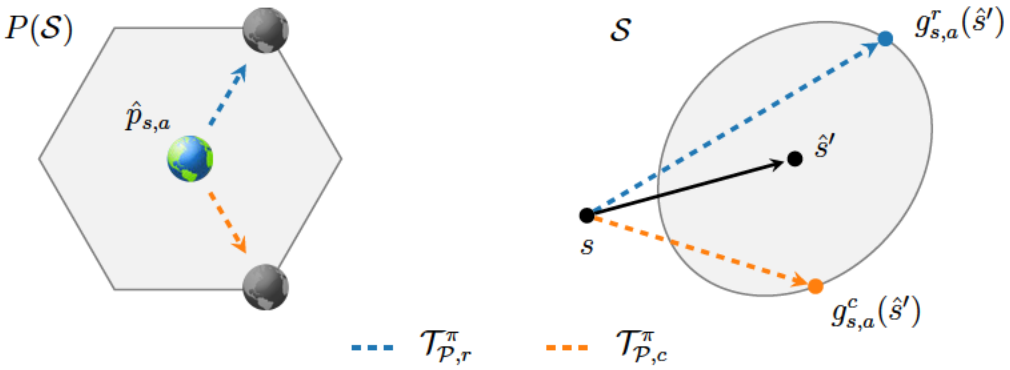

Figure 5·2: Illustration of tractable reformulation in Theorem 5.1. Left: Optimal transport uncertainty set in $P(\mathcal{S})$, along with worst-case transition models corresponding to robust Bellman operators. Right: Optimal Transport Perturbations in $\mathcal{S}$ for a given next state sample $\hat{s}' \sim \hat{p}_{s,a}$. The black arrow denotes the state transition observed in the nominal environment, and the dashed arrows denote virtual transitions used only to calculate robust Bellman operators.

(Pinto et al., 2017; Tessler et al., 2019a; Vinitsky et al., 2020), our Optimal Transport Perturbations have no impact on trajectory rollouts in the nominal training environment. Instead, these perturbations are applied after data collection to construct worst-case virtual state transitions.

5.4 Deep Perturbation Networks

From Theorem 5.1, we can calculate Bellman targets for our robust Q functions $Q^\pi_{\mathcal{P},r}(s,a)$ and $Q^\pi_{\mathcal{P},c}(s,a)$ by considering adversarially perturbed versions of next states sampled from $\hat{p}_{s,a}$. We can construct these adversarial perturbations by solving (5.7) and (5.8), respectively. Note that the perturbation functions $g^r_{s,a}, g^c_{s,a} \in \mathcal{G}$ from Theorem 5.1 differ across state-action pairs. We can combine these perturbation functions across state-action pairs by including (s,a) as input for context, in addition to the next state $\hat{s}'$ to be perturbed. This leads to the combined perturbation functions $g^r, g^c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{S}$, where $g^r(s,a,\hat{s}') = g^r_{s,a}(\hat{s}')$ and $g^c(s,a,\hat{s}') = g^c_{s,a}(\hat{s}')$. We let

$\mathcal{F}$ be the set of all functions from $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ to $\mathcal{S}$, with $g^r, g^c \in \mathcal{F}$.

In the context of deep RL, we consider a class of perturbation functions $\mathcal{F}_\delta \subseteq \mathcal{F}$ parameterized by a neural network $\delta : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{S}$. In our experiments, we consider tasks where $\mathcal{S} = \mathbb{R}^m$ and we apply multiplicative perturbations to state transitions. In particular, we consider perturbation functions of the form

$$g_\delta(s, a, \hat{s}') = s + (\hat{s}' - s)(1 + \delta(s, a, \hat{s}')), \tag{5.9}$$

where $\delta(s, a, \hat{s}') \in \mathbb{R}^m$ and all operations are performed per-coordinate. By defining $\mathcal{F}_\delta$ in this way, we obtain plausible adversarial transitions that are interpretable, where $\delta(s, a, \hat{s}')$ represents the percentage change to the nominal state transition in each coordinate.

Using $d_{s,a}$ from (5.4), we have that

$$d_{s,a}(\hat{s}', g_\delta(s, a, \hat{s}')) = \frac{1}{m}\|\delta(s, a, \hat{s}')\|_2^2.$$

Then, following common practice in off-policy deep RL, we combine the perturbation function updates in (5.7) and (5.8) across state-action pairs by averaging over samples from the replay buffer. By doing so, we can efficiently update our perturbation networks in a deep RL setting according to

$$\delta_r \in \arg\min_\delta \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} [V_r^\pi(g_\delta(s, a, \hat{s}'))] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} \left[\|\delta(s, a, \hat{s}')\|_2^2\right] \leq m\epsilon_\delta^2, \tag{5.10}$$

$$\delta_c \in \arg\max_\delta \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} [V_c^\pi(g_\delta(s, a, \hat{s}'))] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} \left[\|\delta(s, a, \hat{s}')\|_2^2\right] \leq m\epsilon_\delta^2, \tag{5.11}$$

where $(s, a, \hat{s}') \sim \mathcal{D}$ are transitions collected in the nominal environment and ϵ_δ represents the average per-coordinate magnitude of $\delta(s, a, \hat{s}')$ with $\mathbb{E}_{(s,a)\sim\mathcal{D}}[\epsilon_{s,a}] = \epsilon_\delta^2$. It is also possible to satisfy perturbation function constraints at every state-action pair through the use of clipping mechanisms, if desired. Note that any violation of the

Algorithm 5.1: Safe RL with Optimal Transport Perturbations

Input: initial policy π_0; critics $Q_{\theta_r}, Q_{\theta_c}$; OTP networks δ_r, δ_c.

for $k = 0, 1, 2, \ldots$ **do**

  Collect data $\tau \sim (\pi_k, \hat{p})$ and store it in $\mathcal{D}$.

  for K *updates* **do**

    Sample a batch of data $(s, a, r, c, \hat{s}') \sim \mathcal{D}$.

    Update OTP networks δ_r, δ_c according to (5.10) and (5.11).

    Calculate Bellman targets in (5.12) and (5.13), and update critics $Q_{\theta_r}, Q_{\theta_c}$ to minimize $\mathcal{L}_{\mathcal{P},r}(\theta_r), \mathcal{L}_{\mathcal{P},c}(\theta_c)$.

    Update policy π according to (5.1).

  end

end

perturbation function constraints in practice will only lead to additional robustness and will not negatively impact safety.

We train separate reward and cost perturbation networks δ_r and δ_c, and we apply the resulting Optimal Transport Perturbations to calculate the Bellman targets in (5.5) and (5.6) for training the robust Q functions $Q^{\pi}_{\mathcal{P},r}(s, a)$ and $Q^{\pi}_{\mathcal{P},c}(s, a)$. For $(s, a, \hat{s}') \sim \mathcal{D}$, we consider the sample-based estimates

$$\hat{\mathcal{T}}^{\pi}_{\mathcal{P},r} Q_r(s, a) = r(s, a) + \gamma V^{\pi}_r(g_{\delta_r}(s, a, \hat{s}')), \tag{5.12}$$

$$\hat{\mathcal{T}}^{\pi}_{\mathcal{P},c} Q_c(s, a) = c(s, a) + \gamma V^{\pi}_c(g_{\delta_c}(s, a, \hat{s}')). \tag{5.13}$$

5.5 Algorithm

We summarize our approach to robust and safe RL in Algorithm 5.1. At every update, we sample previously collected data from a replay buffer $\mathcal{D}$. We update our reward and cost perturbation networks δ_r and δ_c according to (5.10) and (5.11), respectively. Then, we estimate Bellman targets according to (5.12) and (5.13), which we use to update our critics via standard temporal difference learning loss functions.

We consider parameterized critics Q_{θ_r} and Q_{θ_c}, and we optimize their parameters to minimize the loss functions

$$\mathcal{L}_{\mathcal{P},r}(\theta_r) = \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}}\left[\left(Q_{\theta_r}(s,a) - \hat{\mathcal{T}}^{\pi}_{\mathcal{P},r}\bar{Q}_{\theta_r}(s,a)\right)^2\right],$$
$$\mathcal{L}_{\mathcal{P},c}(\theta_c) = \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}}\left[\left(Q_{\theta_c}(s,a) - \hat{\mathcal{T}}^{\pi}_{\mathcal{P},c}\bar{Q}_{\theta_c}(s,a)\right)^2\right],$$

where $\bar{Q}_{\theta_r}$ and $\bar{Q}_{\theta_c}$ represent target critic networks. Finally, we use these critic estimates to update our policy according to (5.1).

Compared to standard safe RL methods, the only additional components of our approach are the perturbation networks used to apply Optimal Transport Perturbations, which we train alongside the critics and the policy using standard gradient-based methods. Otherwise, the computations for updating the critics and policy remain unchanged. Therefore, it is simple to incorporate our OTP framework into existing deep safe RL algorithms in order to provide robustness guarantees on performance and safety.

5.6 Experiments

We analyze the use of Optimal Transport Perturbations for robust and safe RL on continuous control tasks with safety constraints in the Real-World RL Suite (Dulac-Arnold et al., 2020, 2021). We consider 5 constrained tasks over 3 domains: Cartpole Swingup, Walker Walk, Walker Run, Quadruped Walk, and Quadruped Run. Each task has a horizon length of 1,000 with $r(s,a) \in [0,1]$ and $c(s,a) \in \{0,1\}$. In all tasks, we consider a safety budget of $B = 100$. A policy incurs cost in the Cartpole domain when the slider moves outside of a specified range, in the Walker domain when the velocity of any joint exceeds a threshold, and in the Quadruped domain when joint angles are outside of an acceptable range. See the Appendix for additional information on the safety constraints we consider, and see Dulac-Arnold et al. (2021) for detailed

definitions.

We train policies in a nominal training environment for 1 million steps over 5 random seeds, and we evaluate the robustness of the learned policies in terms of both performance and safety across a range of perturbed test environments via 10 trajectory rollouts. We define these test environments by perturbing a simulator parameter in each domain. We vary the length of the pole in the Cartpole domain, the length of the torso in the Walker domain, and the density of the torso in the Quadruped domain. Note that the parameter value associated with the nominal training environment is in the center of the range of parameter values considered at test time. See the Appendix for additional information on the environment perturbations considered for each task.

Our Optimal Transport Perturbations can be combined with many popular safe RL algorithms, which is a benefit of our methodology. In our experiments, we consider the safe RL algorithm Constraint-Rectified Policy Optimization (CRPO) (Xu et al., 2021), and we use the unconstrained deep RL algorithm Maximum a Posteriori Policy Optimization (MPO) (Abdolmaleki et al., 2018) to calculate policy updates in CRPO. For a fair comparison, we apply CRPO with MPO policy updates as the baseline safe RL algorithm in every method we consider in our experiments. We train a multivariate Gaussian policy, where the mean and diagonal covariance at a given state are parameterized by a neural network. We also consider separate neural network parameterizations for the reward and cost critics. See the Appendix for additional implementation details, including network architectures and values of all hyperparameters.[1]

We incorporate robustness into the baseline safe RL algorithm in three ways: (i) Optimal Transport Perturbations, (ii) adversarial RL using the action-robust PR-MDP framework from Tessler et al. (2019a) applied to the safety constraint, and

[1]Code is publicly available at `https://github.com/jqueeney/robust-safe-rl`.

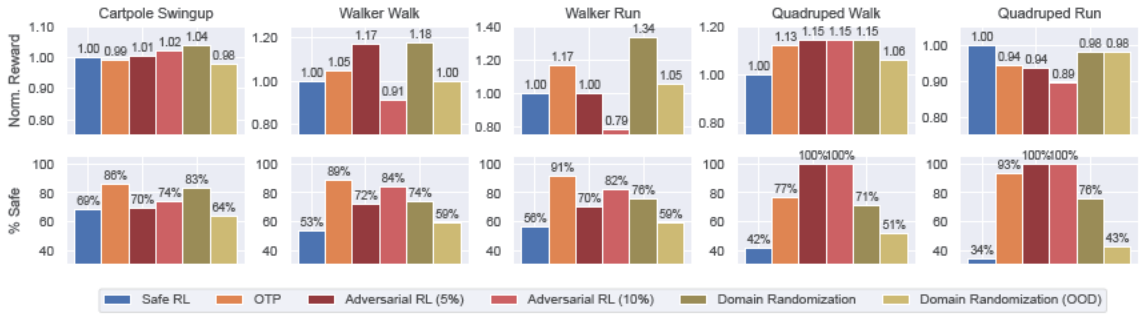

Figure 5·3: Performance summary by task, aggregated across test environments. Performance of adversarial RL is evaluated without adversarial interventions. Top: Average total reward, normalized relative to the average performance of standard safe RL for each test environment. Bottom: Percentage of policies that satisfy the safety constraint across all test environments.

(iii) the soft-robust approach of domain randomization (Peng et al., 2018). For our Optimal Transport Perturbations, we consider $\epsilon_\delta = 0.02$ (i.e., 2% perturbations on average). Figure 5·3 and Table 5.1 summarize the performance of all algorithms at deployment time, where performance metrics are aggregated across the range of perturbed test environments. See the Appendix for detailed results across all test environments. Note that the training process of each algorithm only considers safety for a specific set of environments, so we do not expect the resulting policies to remain safe across all possible test cases.

5.6.1 Comparison to Safe Reinforcement Learning

Figure 5·4 compares our OTP framework to standard safe RL across all tasks and perturbed test environments. By applying Optimal Transport Perturbations to the objective and constraint in safe RL, we achieve meaningful test-time improvements compared to the standard non-robust version of safe RL. While in most cases we observe a decrease in total rewards in the nominal environment in order to achieve robustness, as expected, on average our framework leads to an increase in total rewards

Table 5.1: Aggregate performance summary.

		Normalized Ave.‡		Rollouts Require*	
Algorithm	% Safe†	Reward	Cost	Adversary	Simulator
Safe RL	51%	1.00	1.00	No	No
OTP	**87%**	**1.06**	**0.34**	**No**	**No**
Adversarial RL (5%)	82%	1.05	0.48	Yes	No
Adversarial RL (10%)	88%	0.95	0.28	Yes	No
Domain Randomization	76%	1.14	0.72	No	Yes
Domain Randomization (OOD)	55%	1.02	1.02	No	Yes

† Percentage of policies that satisfy the safety constraint across all tasks and test environments.
‡ Normalized relative to the average performance of standard safe RL for each task and test environment.
* Denotes need for adversary or simulator during data collection (i.e., trajectory rollouts) for training.

of 1.06x relative to safe RL across the range of test environments. Most importantly, we see a significant improvement in safety, with our algorithm satisfying constraints in 87% of test cases (compared to 51% for safe RL) and incurring 0.34x the costs of safe RL, on average. Note that we achieve this robustness while collecting data from the same training environment considered in standard safe RL, without requiring adversarial interventions in the environment or domain knowledge on the structure of the perturbed test environments.

5.6.2 Comparison to Adversarial Reinforcement Learning

Next, we compare our approach to the PR-MDP framework (Tessler et al., 2019a), an adversarial RL method that randomly applies adversarial actions a percentage of the time during training. In order to apply this method to the safe RL setting, we train the adversary to maximize costs. We consider 5% and 10% probabilities of intervention during training. As shown in Figure 5·3, this adversarial approach leads to robust constraint satisfaction at test time in the Quadruped tasks. Our OTP framework, on the other hand, leads to improved constraint satisfaction in the remaining 3 out of 5 tasks. Note that the robust safety demonstrated by the more adversarial

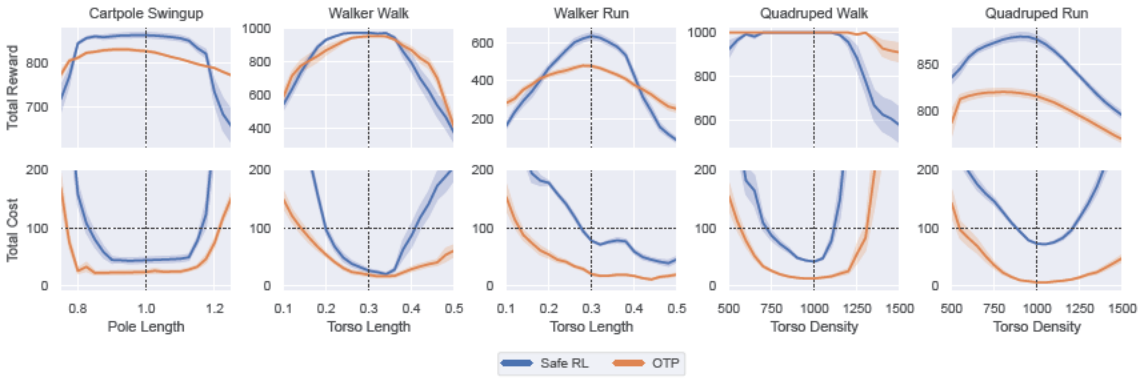

Figure 5·4: Comparison of OTP with standard safe RL across tasks and test environments. Shading denotes half of one standard error across policies. Vertical dotted lines represent nominal training environment. Top: Total reward. Bottom: Total cost, where horizontal dotted lines represent the safety budget and values below these lines represent safety constraint satisfaction.

implementation (10% probability of intervention) also results in lower total rewards on average, and is the only robust approach in Table 5.1 that underperforms standard safe RL in terms of reward. The less adversarial implementation (5% probability of intervention) is comparable to our OTP framework in terms of total rewards, but leads to a decrease in safety constraint satisfaction to 82%. Therefore, our OTP formulation demonstrates the safety benefits of the more adversarial setting and the reward benefits of the less adversarial setting.

In addition, an important drawback of adversarial RL is that it requires the intervention of an adversary in the training environment. Therefore, in order to achieve robust safety at deployment time, adversarial RL incurs additional cost during training due to the presence of an adversary. Even in the Quadruped tasks where adversarial RL results in near-zero cost at deployment time, Figure 5·5 shows that adversarial RL leads to the highest total cost during training due to adversarial interventions. In many real-world situations, this additional cost during training is undesirable. Our OTP framework, on the other hand, achieves the lowest total cost during training,

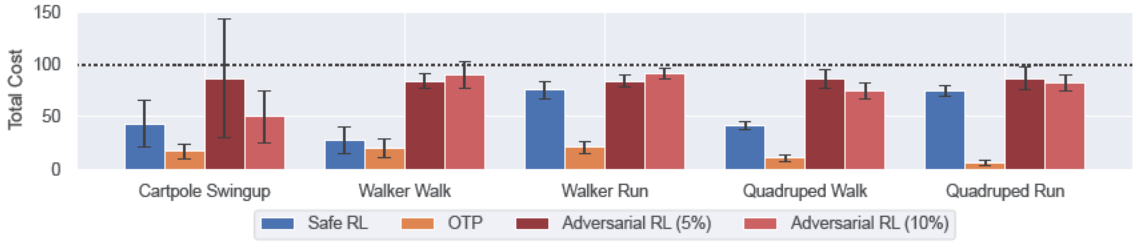

Figure 5·5: Average final training cost in the nominal training environment. Training cost of adversarial RL includes impact of adversarial interventions. Black bars denote one standard deviation across policies. Horizontal dotted line represents safety budget.

while also resulting in robust safety when deployed in perturbed environments. This is due to the fact that our Optimal Transport Perturbations do not impact the data collection process during training.

5.6.3 Comparison to Domain Randomization

Finally, we compare our OTP framework to the soft-robust approach of domain randomization (Peng et al., 2018), which assumes access to a range of environments during training through the use of a simulator. By training across half of the test environments, domain randomization achieves strong performance across test cases in terms of reward (1.14x compared to safe RL, on average), which was the motivation for its development in the setting of sim-to-real transfer. However, domain randomization only satisfies safety constraints in 76% of test cases, which is lower than both OTP and adversarial RL which explicitly consider robust formulations. This is likely due to its soft-robust approach that focuses on average performance across the training distribution.

It is important to note that domain randomization not only requires access to a range of training environments, it also requires prior knowledge on the structure of potential disturbances to define its training distribution. In order to evaluate the case

where we lack domain knowledge, we consider an out-of-distribution (OOD) version of domain randomization that is trained on a distribution over a different parameter than the one varied at test time. When the training distribution is not appropriately selected, we see that domain randomization provides little benefit compared to standard safe RL. Our OTP framework, on the other hand, guarantees robust and safe performance under general forms of model uncertainty while only collecting data from a single training environment.

5.7 Summary

In this chapter, we have developed a general framework for robust and safe RL through the use of an optimal transport uncertainty set. We demonstrated that we can guarantee robustness to general forms of environment disturbances by applying adversarial perturbations to observed state transitions. These Optimal Transport Perturbations can be efficiently implemented by constructing virtual transitions without impacting data collection during training, and can be easily combined with existing techniques for safe RL to provide protection against unknown disturbances. Because our framework makes limited assumptions on the data collection process during training and does not require directly modifying the environment, it should be compatible with many real-world decision making applications.

Note that our use of the RC-MDP framework guarantees robust, safe performance for all environments in the uncertainty set $\mathcal{P}$. Therefore, we must specify the transport cost function $d_{s,a}$ and radius $\epsilon_{s,a}$ to achieve the desired robustness without being overly conservative. An alternative method for incorporating model uncertainty is to instead consider a distribution over possible transition models. We apply this approach in the next chapter.

Chapter 6

Risk-Averse Model Uncertainty with Safety Constraints

The previous chapter incorporated model uncertainty into safe RL by considering worst-case transition models in an uncertainty set. In this chapter, we instead represent model uncertainty using a distribution over possible transition models, which provides benefits compared to the common robust RL approach based on an uncertainty set. First, a distribution represents a more informative way to represent model uncertainty, as it incorporates additional prior knowledge about which transition models are more likely than others at deployment time. In addition, the worst-case approach of robust RL with an uncertainty set introduces complex minimax optimization problems throughout training. These difficult minimax formulations can be avoided by instead optimizing for average performance over a distribution of possible transition models. However, by focusing on average performance, this popular approach to model uncertainty lacks robustness guarantees. In order to incorporate robustness while considering a distribution over transition models, we introduce a *risk-averse perspective towards model uncertainty* through the use of coherent distortion risk measures.

6.1 Coherent Distortion Risk Measures

Consider a random variable $Z \in \mathcal{Z}$, where $\mathcal{Z}$ is a space of random variables defined on a given probability space. A real-valued risk measure $\rho : \mathcal{Z} \to \mathbb{R}$ summarizes a

random variable as a value on the real line. In this section, we consider cost random variables where a lower value of $\rho(Z)$ is better. We can define a corresponding risk measure ρ^+ for reward random variables through an appropriate change in sign, where $\rho^+(Z) = -\rho(-Z)$.

Recently, Majumdar and Pavone (2020) proposed a set of axioms to characterize desirable properties of risk measures in the context of robotics.

Definition 6.1 (Risk Measure Axioms)**.** *Majumdar and Pavone (2020) proposed a set of* risk measure axioms *defined as follows:*

A1. Monotonicity: $\rho(Z) \leq \rho(Z')$ *if* $Z, Z' \in \mathcal{Z}$ *and* $Z \leq Z'$ *almost everywhere.*

A2. Translation invariance: $\rho(Z + \alpha) = \rho(Z) + \alpha$ *if* $\alpha \in \mathbb{R}$ *and* $Z \in \mathcal{Z}$*.*

A3. Positive homogeneity: $\rho(\tau Z) = \tau\rho(Z)$ *if* $\tau \geq 0$ *and* $Z \in \mathcal{Z}$*.*

A4. Convexity: $\rho(\lambda Z + (1-\lambda)Z') \leq \lambda\rho(Z) + (1-\lambda)\rho(Z')$ *if* $\lambda \in [0, 1]$ *and* $Z, Z' \in \mathcal{Z}$*.*

A5. Comonotonic additivity: $\rho(Z+Z') = \rho(Z)+\rho(Z')$ *if* $Z, Z' \in \mathcal{Z}$ *are comonotonic.*

A6. Law invariance: $\rho(Z) = \rho(Z')$ *if* $Z, Z' \in \mathcal{Z}$ *are identically distributed.*

See Majumdar and Pavone (2020) for a discussion on the intuition behind these axioms. Risk-sensitive methods typically focus on classes of risk measures that satisfy some or all of these axioms, such as coherent risk measures (Artzner et al., 1999) and distortion risk measures (Wang, 1996; Dhaene et al., 2012).

Definition 6.2 (Coherent Risk Measure)**.** *A risk measure* ρ *is a* coherent risk measure *if it satisfies Axioms A1–A4.*

Definition 6.3 (Distortion Risk Measure)**.** *Let* $g : [0, 1] \to [0, 1]$ *be a non-decreasing, left-continuous function with* $g(0) = 0$ *and* $g(1) = 1$*. A* distortion risk measure *with respect to* g *is defined as*

$$\rho(Z) = \int_0^1 F_Z^{-1}(u) d\tilde{g}(u),$$

where F_Z^{-1} *is the inverse cumulative distribution function of* Z *and* $\tilde{g}(u) = 1 - g(1-u)$*.*

All distortion risk measures satisfy Axioms A1–A3 and Axioms A5–A6. A distortion risk measure is coherent if and only if g is concave (Wirch and Hardy, 2003). In this chapter, we focus on the class of coherent distortion risk measures, which satisfy all of the axioms proposed in Majumdar and Pavone (2020). Many commonly used risk measures belong to this class, including expectation, conditional value-at-risk (CVaR), and the Wang transform (Wang, 2000) for $\eta \geq 0$ which is defined by the distortion function $g_\eta(u) = \Phi(\Phi^{-1}(u) + \eta)$, where Φ is the standard Normal cumulative distribution function.

Risk-sensitive methods typically focus on the *aleatoric uncertainty* in RL, which refers to the range of stochastic outcomes within a single MDP. Rather than considering the standard expected value objective, they learn risk-sensitive policies over this distribution of possible outcomes in a fixed MDP (Shen et al., 2014; Chow et al., 2015; Tamar et al., 2015; Bellemare et al., 2017; Dabney et al., 2018; Ma et al., 2020; Keramati et al., 2020; L.A. and Fu, 2022). We also consider the use of risk measures in this chapter, but different from standard risk-sensitive RL methods we apply a risk measure over model uncertainty instead of aleatoric uncertainty.

6.2 Risk-Averse Model Uncertainty

In the standard safe RL setting with fixed transition model p, the policy update in (2.6) depends on $Q^\pi_{p,r}(s,a)$ and $Q^\pi_{p,c}(s,a)$. These Q functions represent fixed points of the standard Bellman operators

$$\mathcal{T}^\pi_{p,r} Q(s,a) := r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} [Q(s',a')] \right],$$
$$\mathcal{T}^\pi_{p,c} Q(s,a) := c(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} [Q(s',a')] \right].$$

In this work, however, we are interested in a distribution over possible transition models given by $\mu = \prod_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mu_{s,a}$ rather than a fixed transition model. The dis-

tribution μ provides a natural way to capture our uncertainty about the unknown transition model at deployment time. Next, we must incorporate this model uncertainty in the definition of our Q functions. Prior methods have done this by applying the expectation operator over $\mu_{s,a}$ at every transition (Derman et al., 2018), which corresponds to the constrained optimization problem in (2.8). Instead, we adopt a risk-averse view towards model uncertainty in order to learn robust and safe policies. We accomplish this by applying a coherent distortion risk measure ρ *with respect to model uncertainty* at every transition. For a given policy π, this results in the *Risk-Averse Model Uncertainty (RAMU)* Q functions defined as

$$Q^{\pi}_{\rho^+,r}(s,a) := r(s,a) + \gamma \underset{p_{s,a}\sim\mu_{s,a}}{\rho^+} \left(\underset{s'\sim p_{s,a}}{\mathbb{E}} \left[\underset{a'\sim\pi(\cdot|s')}{\mathbb{E}} \left[r(s',a') + \gamma \underset{p_{s',a'}\sim\mu_{s',a'}}{\rho^+} (\dots) \right] \right] \right),$$
$$Q^{\pi}_{\rho,c}(s,a) := c(s,a) + \gamma \underset{p_{s,a}\sim\mu_{s,a}}{\rho} \left(\underset{s'\sim p_{s,a}}{\mathbb{E}} \left[\underset{a'\sim\pi(\cdot|s')}{\mathbb{E}} \left[c(s',a') + \gamma \underset{p_{s',a'}\sim\mu_{s',a'}}{\rho} (\dots) \right] \right] \right),$$

with the use of ρ^+ in $Q^{\pi}_{\rho^+,r}(s,a)$ to account for reward random variables. The notation $\rho_{p_{s,a}\sim\mu_{s,a}}(\,\cdot\,)$ makes clear the fact that the stochasticity of the random variable is with respect to the transition model sampled from $\mu_{s,a}$. Note that we still apply expectations over the aleatoric uncertainty of the C-MDP (i.e., the randomness associated with a stochastic transition model and stochastic policy), while being risk-averse with respect to model uncertainty.

We can write the RAMU Q functions recursively as

$$Q^{\pi}_{\rho^+,r}(s,a) = r(s,a) + \gamma \underset{p_{s,a}\sim\mu_{s,a}}{\rho^+} \left(\underset{s'\sim p_{s,a}}{\mathbb{E}} \left[\underset{a'\sim\pi(\cdot|s')}{\mathbb{E}} \left[Q^{\pi}_{\rho^+,r}(s',a') \right] \right] \right),$$
$$Q^{\pi}_{\rho,c}(s,a) = c(s,a) + \gamma \underset{p_{s,a}\sim\mu_{s,a}}{\rho} \left(\underset{s'\sim p_{s,a}}{\mathbb{E}} \left[\underset{a'\sim\pi(\cdot|s')}{\mathbb{E}} \left[Q^{\pi}_{\rho,c}(s',a') \right] \right] \right).$$

These recursive definitions motivate corresponding RAMU Bellman operators.

Definition 6.4 (RAMU Bellman Operators)**.** *For a given policy π, the* RAMU Bell-

man operators *are defined as*

$$\mathcal{T}^{\pi}_{\rho^+,r}Q(s,a) := r(s,a) + \gamma \underset{p_{s,a}\sim\mu_{s,a}}{\rho^+} \left(\mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a'\sim\pi(\cdot|s')} [Q(s',a')] \right] \right),$$
$$\mathcal{T}^{\pi}_{\rho,c}Q(s,a) := c(s,a) + \gamma \underset{p_{s,a}\sim\mu_{s,a}}{\rho} \left(\mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a'\sim\pi(\cdot|s')} [Q(s',a')] \right] \right).$$

Note that the RAMU Bellman operators can also be interpreted as applying a coherent distortion risk measure over standard Bellman targets, which are random variables with respect to the transition model $p_{s,a} \sim \mu_{s,a}$ for a given state-action pair.

Lemma 6.1. *The RAMU Bellman operators can be written in terms of standard Bellman operators as*

$$\mathcal{T}^{\pi}_{\rho^+,r}Q(s,a) = \underset{p_{s,a}\sim\mu_{s,a}}{\rho^+} \left(\mathcal{T}^{\pi}_{p,r}Q(s,a) \right), \quad \mathcal{T}^{\pi}_{\rho,c}Q(s,a) = \underset{p_{s,a}\sim\mu_{s,a}}{\rho} \left(\mathcal{T}^{\pi}_{p,c}Q(s,a) \right). \tag{6.1}$$

Proof. Starting from the definition of $\mathcal{T}^{\pi}_{\rho,c}$ in Definition 6.4, we have that

$$\begin{aligned}
\mathcal{T}^{\pi}_{\rho,c}Q(s,a) &= c(s,a) + \gamma \underset{p_{s,a}\sim\mu_{s,a}}{\rho} \left(\mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a'\sim\pi(\cdot|s')} [Q(s',a')] \right] \right) \\
&= \underset{p_{s,a}\sim\mu_{s,a}}{\rho} \left(c(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a'\sim\pi(\cdot|s')} [Q(s',a')] \right] \right) \\
&= \underset{p_{s,a}\sim\mu_{s,a}}{\rho} \left(\mathcal{T}^{\pi}_{p,c}Q(s,a) \right),
\end{aligned}$$

where the second equality follows from the positive homogeneity (Axiom A3) and translation invariance (Axiom A2) of coherent distortion risk measures, and the final equality follows from the definition of the standard cost Bellman operator $\mathcal{T}^{\pi}_{p,c}$. Similarly, we can apply these steps starting from $\mathcal{T}^{\pi}_{\rho^+,r}Q(s,a)$. □

In the next section, we show that $\mathcal{T}^{\pi}_{\rho^+,r}$ and $\mathcal{T}^{\pi}_{\rho,c}$ are contraction operators, so we can apply standard temporal difference learning techniques to learn the RAMU Q functions $Q^{\pi}_{\rho^+,r}(s,a)$ and $Q^{\pi}_{\rho,c}(s,a)$. Then, by replacing the standard Q functions in (2.6) with RAMU Q functions, we can learn a safe policy that is risk-averse to model

uncertainty by iteratively optimizing

$$\max_{\pi} \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[Q^{\pi_k}_{\rho^+,r}(s,a)\right]\right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[Q^{\pi_k}_{\rho,c}(s,a)\right]\right] \leq B. \tag{6.2}$$

6.3 Robustness Guarantees

Intuitively, our risk-averse perspective places more emphasis on potential transition models that result in higher costs or lower rewards under the current policy, which should result in learning safe policies that are robust to model uncertainty. Next, we formalize the robustness guarantees of our RAMU framework by showing it is equivalent to solving the distributionally robust safe RL problem in (2.9) for appropriate choices of ambiguity sets.

Theorem 6.1. *The RAMU Bellman operators $\mathcal{T}^{\pi}_{\rho^+,r}$ and $\mathcal{T}^{\pi}_{\rho,c}$ are equivalent to distributionally robust Bellman operators with ambiguity sets $\mathcal{U}^+ = \bigotimes_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{U}^+_{s,a}$ and $\mathcal{U} = \bigotimes_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{U}_{s,a}$, respectively, where*

$$\mathcal{T}^{\pi}_{\rho^+,r}Q(s,a) = \inf_{\beta_{s,a}\in\mathcal{U}^+_{s,a}} \mathop{\mathbb{E}}_{p_{s,a}\sim\beta_{s,a}}\left[\mathcal{T}^{\pi}_{p,r}Q(s,a)\right], \tag{6.3}$$

$$\mathcal{T}^{\pi}_{\rho,c}Q(s,a) = \sup_{\beta_{s,a}\in\mathcal{U}_{s,a}} \mathop{\mathbb{E}}_{p_{s,a}\sim\beta_{s,a}}\left[\mathcal{T}^{\pi}_{p,c}Q(s,a)\right], \tag{6.4}$$

and

$$\mathcal{U}^+_{s,a}, \mathcal{U}_{s,a} \subseteq \{\beta_{s,a} \in P(\mathcal{M}) \mid \beta_{s,a} = \xi_{s,a}\mu_{s,a}\}$$

are sets of feasible reweightings of $\mu_{s,a}$ with $\xi_{s,a}$ that depend on the choice of ρ^+ and ρ, respectively.

Proof. Using duality results for coherent risk measures (Shapiro et al., 2014), we see that the application of ρ^+ and ρ at every timestep are equivalent to solving distributionally robust optimization problems over the ambiguity sets of distributions $\mathcal{U}^+$ and $\mathcal{U}$, respectively. This can be interpreted as adversarially reweighting $\mu_{s,a}$ with $\xi_{s,a}$ at every state-action pair. See the Appendix for details. □

Because $\mathcal{T}^{\pi}_{\rho^+,r}$ and $\mathcal{T}^{\pi}_{\rho,c}$ are equivalent to distributionally robust Bellman operators according to Theorem 6.1, they are also equivalent to robust Bellman operators.

Corollary 6.1. *The RAMU Bellman operators $\mathcal{T}^{\pi}_{\rho^+,r}$ and $\mathcal{T}^{\pi}_{\rho,c}$ are equivalent to robust Bellman operators for appropriate choices of uncertainty sets over mixture transition models in $\mathcal{M} = P(\mathcal{S})$.*

Proof. Given the equivalence to distributionally robust Bellman operators as shown in Theorem 6.1, the proof follows by applying results from Xu and Mannor (2010) and Yu and Xu (2016). We include a detailed proof in the Appendix for completeness. □

In addition, given the equivalence established in Theorem 6.1, we can leverage existing results for distributionally robust Bellman operators to show that $\mathcal{T}^{\pi}_{\rho^+,r}$ and $\mathcal{T}^{\pi}_{\rho,c}$ are contraction operators.

Corollary 6.2. *The RAMU Bellman operators $\mathcal{T}^{\pi}_{\rho^+,r}$ and $\mathcal{T}^{\pi}_{\rho,c}$ are γ-contractions in the sup-norm.*

Proof. The proof follows from previous results on distributionally robust Bellman operators (Xu and Mannor, 2010; Yu and Xu, 2016) and robust Bellman operators (Iyengar, 2005; Nilim and Ghaoui, 2005) due to the equivalences shown in Theorem 6.1 and Corollary 6.1. We include a detailed proof in the Appendix for completeness. □

Therefore, we have that $Q^{\pi}_{\rho^+,r}(s,a)$ and $Q^{\pi}_{\rho,c}(s,a)$ can be interpreted as distributionally robust Q functions by Theorem 6.1 (or robust Q functions by Corollary 6.1), and we can apply standard temporal difference methods to learn these RAMU Q functions as a result of Corollary 6.2. Importantly, the results in Theorem 6.1 demonstrate the robustness properties of our RAMU framework, *but they are not used to implement our approach.* Directly implementing (6.3) and (6.4) would require solving for adversarial distributions over transition models at every transition. Instead, our framework provides the same robustness, but the use of risk measures leads to an efficient deep RL implementation as we describe in the following section.

6.4 Efficient Model-Free Implementation

The RAMU policy update in (6.2) takes the same form as the standard safe RL update in (2.6), except for the use of $Q^{\pi}_{\rho^+,r}(s,a)$ and $Q^{\pi}_{\rho,c}(s,a)$. Because our RAMU

Bellman operators are contractions, we can learn these RAMU Q functions by applying standard temporal difference loss functions that are used throughout deep RL. In particular, we consider parameterized critics Q_{θ_r} and Q_{θ_c}, and we optimize their parameters during training to minimize the loss functions

$$\mathcal{L}_{\rho^+,r}(\theta_r) = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\left[\left(Q_{\theta_r}(s,a) - \hat{\mathcal{T}}^{\pi}_{\rho^+,r}\bar{Q}_{\theta_r}(s,a)\right)^2\right],$$
$$\mathcal{L}_{\rho,c}(\theta_c) = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\left[\left(Q_{\theta_c}(s,a) - \hat{\mathcal{T}}^{\pi}_{\rho,c}\bar{Q}_{\theta_c}(s,a)\right)^2\right],$$

where $\hat{\mathcal{T}}^{\pi}_{\rho^+,r}$ and $\hat{\mathcal{T}}^{\pi}_{\rho,c}$ represent sample-based estimates of the RAMU Bellman operators applied to target Q functions denoted by $\bar{Q}$. Therefore, we must be able to efficiently estimate the RAMU Bellman targets, which involve calculating coherent distortion risk measures that depend on the distribution $\mu_{s,a}$.

6.4.1 Sample-Based Estimation of Risk Measures

Using the formulation of our RAMU Bellman operators from Lemma 6.1, we can leverage properties of distortion risk measures to efficiently estimate the results in (6.1) using sample-based weighted averages of standard Bellman targets. For n transition models $p^{(i)}_{s,a}$, $i = 1, \ldots, n$, sampled independently from $\mu_{s,a}$ and sorted according to their corresponding Bellman targets, consider the weights

$$w^{(i)}_{\rho} = g\left(\frac{i}{n}\right) - g\left(\frac{i-1}{n}\right),$$

where g defines the distortion risk measure ρ according to Definition 6.3. See Figure 6·1 for the weights associated with the risk measures used in our experiments. Then, from Jones and Zitikis (2003) we have that

$$\sum_{i=1}^{n} w^{(i)}_{\rho^+}\mathcal{T}^{\pi}_{p^{(i)},r}Q(s,a), \quad \sum_{i=1}^{n} w^{(i)}_{\rho}\mathcal{T}^{\pi}_{p^{(i)},c}Q(s,a),$$

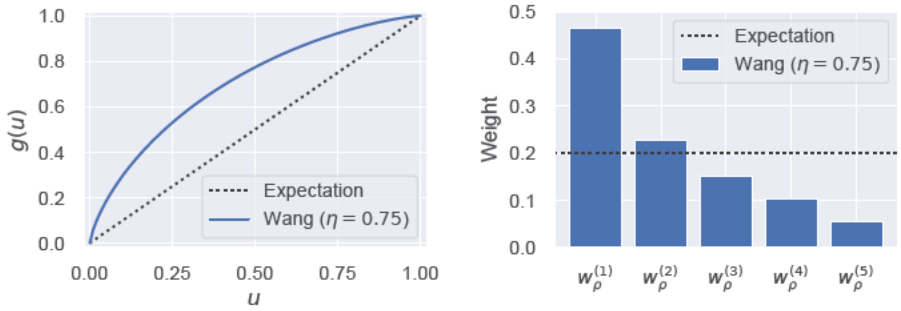

Figure 6·1: Coherent distortion risk measures used in RAMU experiments. Left: Distortion function g. Right: Weights for sample-based estimates in (6.5) when $n = 5$.

are consistent estimators of the results in (6.1), where $\mathcal{T}^{\pi}_{p^{(i)},r}Q(s,a)$ are sorted in ascending order and $\mathcal{T}^{\pi}_{p^{(i)},c}Q(s,a)$ are sorted in descending order. Finally, we can replace $\mathcal{T}^{\pi}_{p^{(i)},r}Q(s,a)$ and $\mathcal{T}^{\pi}_{p^{(i)},c}Q(s,a)$ with the standard unbiased sample-based estimates

$$\hat{\mathcal{T}}^{\pi}_{p^{(i)},r}Q(s,a) = r(s,a) + \gamma Q(s',a'), \quad \hat{\mathcal{T}}^{\pi}_{p^{(i)},c}Q(s,a) = c(s,a) + \gamma Q(s',a'),$$

where $s' \sim p^{(i)}_{s,a}$ and $a' \sim \pi(\cdot \mid s')$. This leads to the sample-based estimates

$$\hat{\mathcal{T}}^{\pi}_{\rho^+,r}Q(s,a) = \sum_{i=1}^{n} w^{(i)}_{\rho^+}\hat{\mathcal{T}}^{\pi}_{p^{(i)},r}Q(s,a), \quad \hat{\mathcal{T}}^{\pi}_{\rho,c}Q(s,a) = \sum_{i=1}^{n} w^{(i)}_{\rho}\hat{\mathcal{T}}^{\pi}_{p^{(i)},c}Q(s,a), \tag{6.5}$$

which we use to train our RAMU Q functions. Note that the estimates in (6.5) can be computed very efficiently, which is a major benefit of our RAMU framework compared to robust RL methods based on uncertainty sets. Next, we describe how we can sample models $p^{(i)}_{s,a}$, $i = 1, \ldots, n$, from $\mu_{s,a}$, and generate state transitions from these models to use in the calculation of our sample-based Bellman targets in (6.5).

6.4.2 Generative Distribution of Transition Models

Note that our framework holds for any choice of distribution μ over transition models. Therefore, it is possible to define μ in a way that leverages detailed simulator access

for sampling transition models $p_{s,a}^{(i)} \sim \mu_{s,a}$ and corresponding next states $s' \sim p_{s,a}^{(i)}$. However, in order for our methods to be broadly applicable to any setting where data can be collected under a single training environment $\hat{p}$, we propose a generative approach to sampling transition models and corresponding next states by defining the distribution μ over perturbed versions of the training environment. We consider a model-free implementation in this work, but model-based approaches are also possible. This choice of distribution μ captures general uncertainty in the training environment, without requiring specific domain knowledge of potential disturbances.

Consider a function $f_x : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ that takes as input a state transition $(s, \hat{s}')$ and outputs a perturbed next state $s' = f_x(s, \hat{s}')$, where f_x depends on a latent variable $x \sim X$. We also define a transition model $p_{s,a}(x)$ for every $x \sim X$ that shifts the probability of $\hat{s}'$ under $\hat{p}_{s,a}$ to $s' = f_x(s, \hat{s}')$. Then, by considering a distribution over latent space, we implicitly define a distribution $\mu_{s,a}$ over transition models $p_{s,a}(x)$. By defining $\mu_{s,a}$ in this way, it becomes easy to generate the next state samples needed to calculate the Bellman target estimates in (6.5). We sample a latent variable, which defines the transition model $p_{s,a}(x)$. Then, we can generate a state transition under this sampled model by simply perturbing the next state observed in the training environment according to f_x. In particular, for data collected in the training environment we have that $\hat{s}' \sim \hat{p}_{s,a}$, so $s' = f_x(s, \hat{s}')$ represents the corresponding sample from the transition model $p_{s,a}(x)$.

In our experiments, we consider a simple implementation for the common case where $\mathcal{S} = \mathbb{R}^m$. We use uniformly distributed latent variables $x \sim U([-2\epsilon, 2\epsilon]^m)$, and we define the perturbation function as

$$f_x(s, \hat{s}') = s + (\hat{s}' - s)(1 + x),$$

where all operations are performed per-coordinate. Note that this represents the

Algorithm 6.1: Safe RL with Risk-Averse Model Uncertainty

Input: initial policy π_0; critics $Q_{\theta_r}, Q_{\theta_c}$; risk measures ρ^+, ρ; latent random variable X.

for $k = 0, 1, 2, \ldots$ **do**

Collect data $\tau \sim (\pi_k, \hat{p})$ and store it in $\mathcal{D}$.

for K *updates* **do**

Sample a batch of data $(s, a, r, c, \hat{s}') \sim \mathcal{D}$.

Sample n latent variables $x_i \sim X$ per data point, and compute next state samples $f_{x_i}(s, \hat{s}')$, $i = 1, \ldots, n$.

Calculate Bellman targets in (6.5), and update critics $Q_{\theta_r}, Q_{\theta_c}$ to minimize $\mathcal{L}_{\rho^+,r}(\theta_r), \mathcal{L}_{\rho,c}(\theta_c)$.

Update policy π according to (6.2).

end

end

same percentage-based perturbation structure considered in Chapter 5. Therefore, the latent variable $x \sim U([-2\epsilon, 2\epsilon]^m)$ can be interpreted as the percentage change in each dimension of a state transition observed in the training environment, where the average magnitude of the percentage change is ϵ. The hyperparameter ϵ determines the distribution $\mu_{s,a}$ over transition models, where a larger value of ϵ leads to transition models that vary more significantly from the training environment. The structure of f_x provides an intuitive, scale-invariant meaning for the hyperparameter ϵ, which makes it easy to tune in practice.

6.5 Algorithm

We summarize the implementation of our RAMU framework in Algorithm 6.1. Given data collected in a single training environment, we can efficiently calculate the sample-based RAMU Bellman targets in (6.5) by (i) sampling from a latent variable $x \sim X$, (ii) computing the corresponding next state samples $f_x(s, \hat{s}')$, and (iii) sorting the

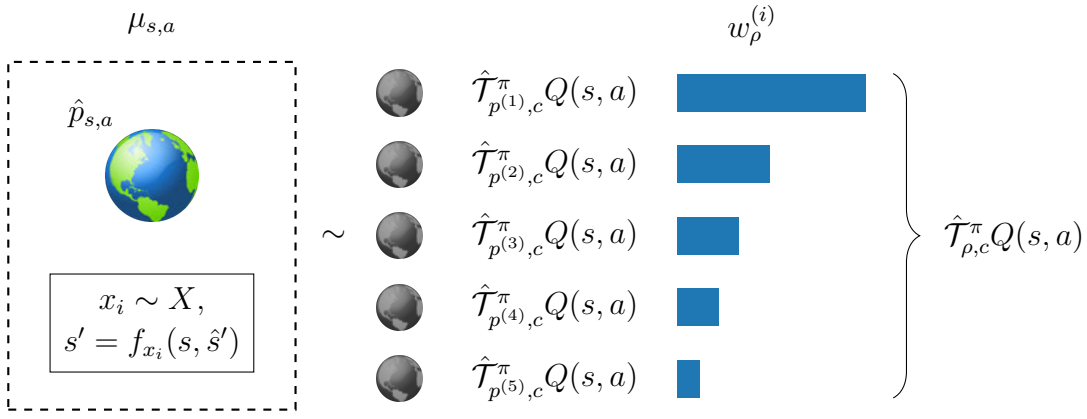

Figure 6·2: Calculation of sample-based RAMU Bellman targets. By sampling n latent variables $x_i \sim X$, we implicitly sample transition models $p_{s,a}(x_i)$ from $\mu_{s,a}$. Then, we compute standard Bellman estimates $\hat{\mathcal{T}}^\pi_{p^{(i)},c} Q(s,a)$ using the corresponding perturbed next state samples given by $f_{x_i}(s, \hat{s}')$. Finally, we sort the standard Bellman estimates in descending order and compute $\hat{\mathcal{T}}^\pi_{\rho,c} Q(s,a)$ using a sample-based weighted average as in (6.5). $\hat{\mathcal{T}}^\pi_{\rho^+,r} Q(s,a)$ can be calculated similarly, with standard Bellman estimates sorted in ascending order.

standard Bellman estimates that correspond to these sampled transition models. See Figure 6·2 for an illustration. Given the sample-based RAMU Bellman targets, updates of the critics and policy have the same form as in standard deep safe RL algorithms. Therefore, our RAMU framework can be easily combined with many popular safe RL algorithms to incorporate model uncertainty with robustness guarantees, using only a minor change to the estimation of Bellman targets that is efficient to implement in practice.

6.6 Comparing Robust and Safe Deployment Frameworks

Both the RAMU framework from this chapter and the Optimal Transport Perturbations framework from Chapter 5 are designed to achieve robust, safe performance in the presence of model uncertainty. In addition, both of these frameworks can be implemented using standard data collection from a single training environment, making

them compatible with settings that require real-world interaction for training. However, these two approaches consider different representations of model uncertainty, which leads to algorithms with distinct characteristics.

Importantly, these two frameworks deliver different types of robustness guarantees. Our OTP framework considers the robust RL setting with an uncertainty set over transition models, providing robustness to worst-case transition models in $\mathcal{M}$. OTP applies robustness around the nominal transition model $\hat{p}$ by training deep perturbation networks to construct worst-case virtual state transitions. Our RAMU framework, on the other hand, is equivalent to a *distributionally robust* RL setting, which instead provides robustness to worst-case *distributions over transition models* in $P(\mathcal{M})$. RAMU applies robustness around the distribution μ through the use of weighted sample averages, without the need to solve minimax optimization problems during training.

In general, the choice between our OTP framework and our RAMU framework should depend on (i) the prior knowledge about model uncertainty at deployment time and (ii) the desired robustness guarantees for the application area of interest. The uncertainty set approach of our OTP formulation provides an intuitive way to guard against general disturbances at deployment time when we have less prior knowledge about model uncertainty, and is more useful when we require stronger robustness guarantees related to worst-case transition models in $\mathcal{M}$. Our RAMU formulation instead considers a distribution μ over transition models, providing a rich representation of model uncertainty that can incorporate additional prior knowledge about which transition models are more likely than others at deployment time. The choice of risk measure ρ also provides flexibility on the level of robustness to apply around the user-defined distribution μ, which can lead to policies with less conservative behavior in settings where distributionally robust guarantees are acceptable.

Table 6.1: Aggregate performance summary.

		Normalized Ave.‡		Rollouts Require*	
Algorithm	% Safe†	Reward	Cost	Adversary	Simulator
Safe RL	51%	1.00	1.00	No	No
RAMU (Wang 0.75)	**80%**	**1.08**	**0.51**	**No**	**No**
RAMU (Expectation)	74%	1.05	0.67	No	No
OTP (Chapter 5)	87%	1.06	0.34	No	No
Adversarial RL (5%)	82%	1.05	0.48	Yes	No
Domain Randomization	76%	1.14	0.72	No	Yes

† Percentage of policies that satisfy the safety constraint across all tasks and test environments.
‡ Normalized relative to the average performance of standard safe RL for each task and test environment.
* Denotes need for adversary or simulator during data collection (i.e., trajectory rollouts) for training.

6.7 Experiments

In order to evaluate the performance and safety of our RAMU framework, we consider the same experimental design used in Chapter 5. In particular, we conduct experiments on the same set of continuous control tasks with safety constraints from the Real-World RL Suite (Dulac-Arnold et al., 2020, 2021), we apply the same safety constraints and safety budget ($B = 100$), and we evaluate performance at deployment time on the same set of perturbed test environments. See Chapter 5 and the Appendix for details.

Our RAMU framework can be combined with several choices of safe RL algorithms. As in Chapter 5, we apply CRPO (Xu et al., 2021) with MPO policy updates (Abdolmaleki et al., 2018) as the baseline safe RL algorithm in every method we consider in our experiments. We also apply the same policy and critic representations used in Chapter 5. We consider a multivariate Gaussian policy with learned mean and diagonal covariance at each state, along with separate reward and cost critics. See the Appendix for implementation details.[1]

[1]Code is publicly available at `https://github.com/jqueeney/robust-safe-rl`.

Figure 6·3: Performance summary by task, aggregated across test environments. Performance of adversarial RL is evaluated without adversarial interventions. Top: Average total reward, normalized relative to the average performance of standard safe RL for each test environment. Bottom: Percentage of policies that satisfy the safety constraint across all test environments.

We summarize the performance and safety of our RAMU framework in Figure 6·3 and Table 6.1, compared to standard safe RL, our OTP framework from Chapter 5, adversarial RL, and domain randomization. We include detailed experimental results across all test environments in the Appendix. We apply our RAMU framework using the Wang transform with $\eta = 0.75$ as the risk measure in both the objective and constraint. In order to understand the impact of being risk-averse to model uncertainty, we also consider the risk-neutral special case of our framework where expectations are applied to the objective and constraint. For our RAMU results, we specify the risk measure in parentheses. Finally, we consider $n = 5$ samples of transition models with latent variable hyperparameter $\epsilon = 0.10$ in order to calculate Bellman targets in our RAMU framework.

6.7.1 Comparison to Safe Reinforcement Learning

First, we analyze the impact of our RAMU framework compared to standard safe RL. In both cases, we train policies using data collected from a single training environment, so the only difference comes from our use of Risk-Averse Model Uncertainty to learn

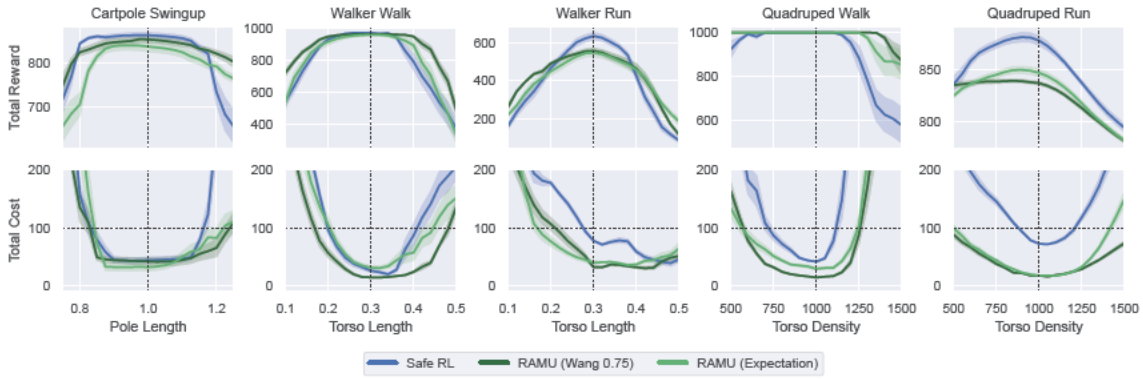

Figure 6·4: Comparison of RAMU with standard safe RL across tasks and test environments. Shading denotes half of one standard error across policies. Vertical dotted lines represent nominal training environment. Top: Total reward. Bottom: Total cost, where horizontal dotted lines represent the safety budget and values below these lines represent safety constraint satisfaction.

RAMU Q functions. Figure 6·4 compares performance and safety across all tasks and test environments. By evaluating the learned policies in perturbed test environments different from the training environment, we see that our RAMU framework provides robustness in terms of both total rewards and safety. In particular, the risk-averse implementation of our algorithm leads to safety constraint satisfaction in 80% of test environments, compared to only 51% with standard safe RL. In addition, this implementation results in higher total rewards (1.08x) and lower total costs (0.51x), on average. We see in Table 6.1 that the use of expectations over model uncertainty (i.e., a risk-neutral approach) also improves robustness in both the objective and constraint, on average, compared to standard safe RL. However, we further improve upon the benefits observed in the risk-neutral case by instead applying a risk-averse perspective.

6.7.2 Comparison to Domain Randomization

Next, we compare our RAMU framework to domain randomization (Peng et al., 2018), a popular approach that also represents model uncertainty using a distribution μ over models. Note that domain randomization considers parametric uncertainty and has the benefit of training on a range of simulated environments, while our method only collects data from a single training environment. As shown in Chapter 5, domain knowledge is critical for defining the training distribution in domain randomization. When the training distribution is not chosen properly, domain randomization provides little benefit compared to standard safe RL. Therefore, we focus our comparison on the case where domain randomization trains across half of the test environments, representing detailed domain knowledge about model uncertainty at deployment time.

With this detailed knowledge, domain randomization achieves the highest normalized average rewards (1.14x) in Table 6.1. However, domain randomization also achieves lower levels of safety, on average, than our risk-averse formulation. In fact, we see in Figure 6·3 that the safety constraint satisfaction of our risk-averse formulation is at least as strong as domain randomization in 4 out of 5 tasks, *despite only training on a single environment with no specific knowledge about the disturbances at test time.* This demonstrates the key benefit of our risk-averse approach to model uncertainty, compared to domain randomization which focuses on average performance.

6.7.3 Comparison to Adversarial Reinforcement Learning

We also compare our approach to adversarial RL using the action-robust PR-MDP framework (Tessler et al., 2019a), which randomly applies worst-case actions a percentage of the time during data collection. Although adversarial RL only collects data from a single training environment, it requires potentially dangerous adversarial interventions during training in order to provide robustness at test time. In order to

apply this method to the safe RL setting, we train an adversary to maximize costs which results in a minimax formulation during training. As shown in Chapter 5, adversarial RL with a 10% probability of intervention leads to overly conservative behavior, so we focus our comparison on adversarial RL with a 5% probability of intervention. Note that the performance of adversarial RL is typically evaluated without adversarial interventions, which requires a clear distinction between training and testing.

We see in Figure 6·3 that adversarial RL learns policies that achieve robust safety constraint satisfaction at test time in the Quadruped tasks. Our risk-averse formulation, on the other hand, achieves higher levels of safety in the remaining 3 out of 5 tasks, and similar levels of safety on average. Unlike adversarial RL, our RAMU framework achieves robust safety in a way that (i) does not alter the data collection process, (ii) does not require training an adversary in a minimax formulation, and (iii) does not require different implementations during training and testing. In addition, our use of a distribution over models represents a less conservative approach than adversarial RL, resulting in higher normalized average rewards as shown in Table 6.1.

6.7.4 Comparison to Optimal Transport Perturbations

Finally, we compare our RAMU framework to our OTP framework introduced in Chapter 5. Note that both frameworks only consider standard data collection from a single training environment. OTP provides robustness by training additional neural networks to construct worst-case virtual state transitions, while RAMU only requires computing weighted sample averages. Figure 6·5 compares the performance and safety of our two frameworks across all tasks and test environments, using the default settings for both approaches. We see that our RAMU framework achieves better performance at test time in terms of total rewards in most cases, resulting in normalized average rewards of 1.08x compared to 1.06x for OTP. Our OTP framework, on

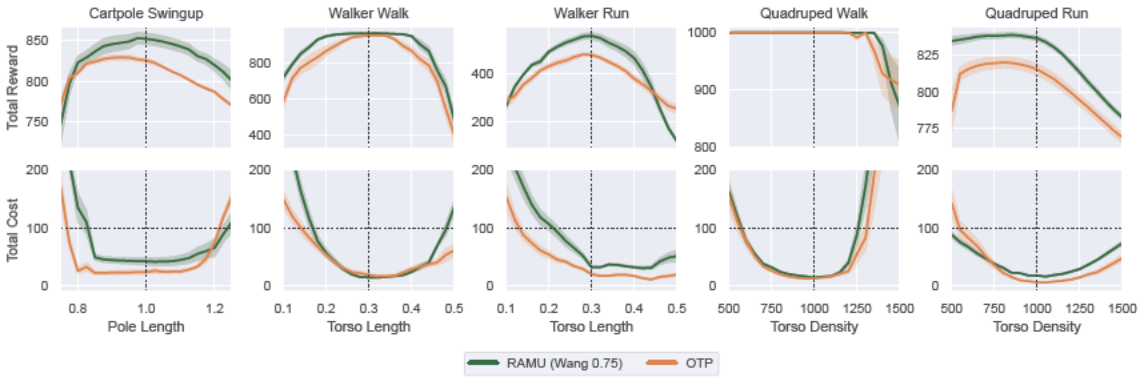

Figure 6·5: Comparison of RAMU with OTP across tasks and test environments. Shading denotes half of one standard error across policies. Vertical dotted lines represent nominal training environment. Top: Total reward. Bottom: Total cost, where horizontal dotted lines represent the safety budget and values below these lines represent safety constraint satisfaction.

the other hand, provides more robust safety, achieving safety constraint satisfaction in 87% of test cases compared to 80% for RAMU. Overall, these experimental results are in line with the different robustness guarantees associated with these frameworks. OTP provides stronger robustness guarantees that lead to better safety constraint satisfaction at deployment time, while RAMU provides less conservative behavior due to its distributionally robust approach. Finally, note that different implementation choices for our OTP and RAMU frameworks will impact the performance comparison between these algorithms.

6.8 Summary

In this chapter, we have presented a framework for safe RL in the presence of model uncertainty that considers a distribution over transition models instead of an uncertainty set. By applying a risk-averse perspective towards model uncertainty, our RAMU framework provides robustness guarantees while still leading to an efficient

deep RL implementation that does not involve minimax optimization problems. As in the previous chapter, this approach only requires data collected from a single training environment. Therefore, it can be used to learn robust, safe policies in real-world domains where fast, high-fidelity simulators are not readily available.

Similar to the OTP framework in Chapter 5, the robustness and safety of our RAMU framework depend on user-defined specifications. The distribution μ defines the uncertainty over transition models, and the risk measure ρ defines the level of robustness to this choice of μ. In addition, note that both our OTP framework and our RAMU framework focus on robustness with respect to model uncertainty. It would also be interesting to extend our techniques to address other forms of uncertainty, such as epistemic uncertainty in model-based RL. We provide a discussion on this future research direction in the next chapter.

Chapter 7

Conclusions and Future Research

Deep RL has the potential to improve operations in important application areas by applying a data-driven, learning-based approach to control. However, deep RL can only benefit society if it can be trusted to produce reliable performance both during training and upon deployment. In this dissertation, we have contributed new techniques to overcome key limitations of existing deep RL methods for real-world decision making and control. In particular, we introduced several *reliable deep RL algorithms* with a focus on (i) stable training from limited data and (ii) robust, safe deployment in the presence of uncertainty.

The first part of the dissertation introduced reliable deep RL algorithms that deliver stable performance throughout training, while also making efficient use of limited data. In Chapter 3 and Chapter 4, we developed techniques to extend the stable training benefits of existing policy improvement methods to the limited data setting. Chapter 3 proposed methods to control the finite-sample estimation error in policy improvement algorithms, and Chapter 4 improved the data efficiency of policy improvement algorithms through theoretically supported sample reuse. These algorithmic contributions are supported by novel policy improvement lower bounds, which guarantee stable training from limited data.

The second part of the dissertation introduced reliable deep RL algorithms that guarantee robust performance and safety in the presence of model uncertainty. In Chapter 5 and Chapter 6, we incorporated safety constraints and model uncertainty

into the deep RL framework. Chapter 5 provided robustness to model uncertainty through the use of an optimal transport uncertainty set over transition models, while Chapter 6 applied a risk-averse perspective to a distribution over transition models. Importantly, these frameworks do not impact the data collection process during training and do not require detailed simulator access. Instead, they can be efficiently implemented using standard data collection from a single training environment, making them compatible with settings that require real-world interaction for training.

Overall, we hope that the contributions of this dissertation provide a foundation for the continued development of reliable deep RL algorithms, with the goal of maximizing the long-term societal impact of deep RL.

7.1 Future Research

In order to unlock the potential of deep RL for decision making and control in application areas of societal importance, it will require continued advances in the area of *reliable deep RL*. We conclude the dissertation with a discussion on potential directions for future research, which build upon the key ideas that we have explored throughout this work.

7.1.1 Stable Training from Large Replay Buffers

The policy improvement algorithms developed in Chapter 3 and Chapter 4 consider on-policy methods as a starting point. As a result, our policy improvement lower bounds only provide support for using data from the current or recent policies during training. These algorithms are useful in limited data settings that require guarantees on stable training, particularly in resource-constrained applications that preclude the use of large replay buffers. However, in settings where large replay buffers are feasible, off-policy deep RL algorithms (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018; Abdolmaleki et al., 2018) have demonstrated strong perfor-

mance on benchmark tasks by reusing a significant amount of previously collected data. Therefore, an interesting avenue for future work includes the development of policy improvement guarantees that are compatible with the aggressive sample reuse in off-policy algorithms. The design of practical performance certificates or verification methods for these algorithms may increase their adoption in real-world control settings where guarantees on stable training are required for deployment.

In many realistic scenarios, the reliable use of large replay buffers for training becomes even more complex than the standard off-policy setting. Real-world systems are often subject to changing environment conditions, involve interactions with other agents that evolve over time, and require the ability to perform multiple tasks. All of these characteristics alter the distribution of data observed during training, and impact the usefulness of past data for future policy updates. Existing methods for multi-agent RL (Foerster et al., 2017; Weber et al., 2022) and continual RL (Isele and Cosgun, 2018; Rolnick et al., 2019) propose heuristic modifications to the replay buffer to address this issue. However, in order to reliably deploy deep RL for data-driven control in non-stationary, multi-agent, and multi-task settings, future research should focus on techniques that provide performance guarantees by selecting the appropriate data to use for training.

7.1.2 Reliable Model-Based Algorithms

We have focused on model-free deep RL algorithms in this dissertation, but model-based methods are a popular choice for improving data efficiency in deep RL. In model-based deep RL, the samples collected from the environment are used to construct a neural network estimate of the transition probability function p of the MDP. This model of the environment can then be used to train the policy, resulting in faster learning that requires less interaction with the true environment. Unfortunately, it can be difficult to learn an accurate model when the environment is complex, and

policy updates tend to exploit model errors which can lead to poor performance and catastrophic failures in the true environment (Deisenroth and Rasmussen, 2011). Existing approaches use ensembles of learned models to address epistemic uncertainty (Chua et al., 2018; Kurutach et al., 2018; Janner et al., 2019; Rajeswaran et al., 2020), but typically lack practical robustness guarantees on performance and safety. The frameworks from Chapter 5 and Chapter 6 represent promising starting points in the design of reliable model-based deep RL algorithms. These frameworks focused on robustness to irreducible model uncertainty, but similar tools could also be applied to the epistemic uncertainty about a learned transition model. By doing so, future research could develop reliable model-based algorithms that are robust to estimation error and guarantee safe, stable performance throughout training.

Appendix A

Detailed Proofs

A.1 Detailed Proofs for Chapter 3

A.1.1 Useful Results

We will make use of the following results in our proofs.

Definition A.1 (Sub-Gaussian Random Variable)**.** *A random variable* $\omega \in \mathbb{R}$ *is* sub-Gaussian *with variance proxy* σ^2 *if* $\mathbb{E}[\omega] = 0$ *and its moment generating function satisfies* $\mathbb{E}[\exp(s\omega)] \leq \exp(s^2\sigma^2/2)$ *for all* $s \in \mathbb{R}$*. We denote this by* $\omega \sim \text{subG}(\sigma^2)$*.*

Definition A.2 (Sub-Gaussian Random Vector)**.** *A random vector* $\boldsymbol{\omega} \in \mathbb{R}^\ell$ *is* sub-Gaussian *with variance proxy* σ^2 *if* $\mathbf{s}'\boldsymbol{\omega} \sim \text{subG}(\sigma^2)$ *for all* $\mathbf{s} \in S^\ell$*, where* $S^\ell = \left\{\mathbf{s} \in \mathbb{R}^\ell \mid \|\mathbf{s}\| = 1\right\}$ *is the unit sphere in* $\mathbb{R}^\ell$*. We denote this by* $\boldsymbol{\omega} \sim \text{subG}_\ell(\sigma^2)$*.*

Lemma A.1. *Let* $\boldsymbol{\omega} \in \mathbb{R}^\ell$ *be a sub-Gaussian random vector with variance proxy* σ^2*. Consider* $\hat{\boldsymbol{\omega}}^{(1)}, \ldots, \hat{\boldsymbol{\omega}}^{(n)}$ *independent, identically distributed random samples of* $\boldsymbol{\omega}$*, with* $\hat{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \hat{\boldsymbol{\omega}}^{(i)}$ *their sample average. Fix* $\alpha \in (0,1)$*. Then,*

$$\mathbb{P}\left(\|\hat{\mathbf{x}}\|_2^2 \leq \sigma^2 R_n^2\right) > 1 - \alpha,$$

where

$$R_n^2 = \frac{1}{n}\left(\ell + 2\sqrt{\ell \log\left(\frac{1}{\alpha}\right)} + 2\log\left(\frac{1}{\alpha}\right)\right).$$

Proof. Consider the random vector $\mathbf{x} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{\omega}^{(i)}$, where $\boldsymbol{\omega}^{(i)}$, $i = 1, \ldots, n$, are independent, identically distributed copies of the random vector $\boldsymbol{\omega}$. Because $\boldsymbol{\omega}^{(i)} \sim \text{subG}_\ell(\sigma^2)$, we have that $\mathbf{x} \sim \text{subG}_\ell(\sigma^2/n)$. Then, the result immediately follows as a special case of Theorem 2.1 in Hsu et al. (2012) applied to the sample $\hat{\mathbf{x}}$ of the random vector $\mathbf{x}$. □

A.1.2 Detailed Proof of Lemma 3.1

Proof. Define $\boldsymbol{\omega} = \boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1/2}(\boldsymbol{\xi}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}}) \in \mathbb{R}^{\ell}$. Note that $\boldsymbol{\omega} \sim \text{subG}_{\ell}(\sigma^2)$ by Assumption 3.1. Therefore, by applying Lemma A.1 with $\hat{\mathbf{x}} = \boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1/2}(\hat{\mathbf{g}}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}})$, we have that

$$\mathbb{P}\left((\hat{\mathbf{g}}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}})'\boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1}(\hat{\mathbf{g}}_{k,\mathbf{Q}} - \mathbf{g}_{k,\mathbf{Q}}) \leq \sigma^2 R_n^2\right) > 1 - \alpha.$$

This implies $\mathbf{g}_{k,\mathbf{Q}} \in \mathcal{U}_k$ with probability at least $1 - \alpha$. □

A.1.3 Detailed Proof of Theorem 3.1

Proof. Consider the function

$$f(\mathbf{u}) = \frac{1}{1-\gamma}\mathbf{u}'\mathbf{y} - \frac{\gamma C^{\pi,\pi_k}}{(1-\gamma)^2}\sqrt{\mathbf{y}'\mathbf{F}_{k,\mathbf{Q}}\mathbf{y}}.$$

Note that we can write the right-hand side of (3.8) as $f(\mathbf{g}_{k,\mathbf{Q}})$, so up to first and second order approximation error we have

$$J(\pi) - J(\pi_k) \geq f(\mathbf{g}_{k,\mathbf{Q}}).$$

By Lemma 3.1, $\mathbf{g}_{k,\mathbf{Q}} \in \mathcal{U}_k$ with probability at least $1 - \alpha$. In this case, we have that $f(\mathbf{g}_{k,\mathbf{Q}}) \geq \min_{\mathbf{u}\in\mathcal{U}_k} f(\mathbf{u})$. Therefore, with probability at least $1 - \alpha$ we have that

$$J(\pi) - J(\pi_k) \geq \min_{\mathbf{u}\in\mathcal{U}_k} f(\mathbf{u}),$$

up to first and second order approximation error.

In order to complete the proof, we now show that $\min_{\mathbf{u}\in\mathcal{U}_k} f(\mathbf{u})$ is equivalent to the right-hand side of (3.9). Note that the second term in the objective $f(\mathbf{u})$ does not depend on $\mathbf{u}$, so we can ignore it when solving the minimization problem over $\mathcal{U}_k$. Omitting this term and ignoring the multiplicative constant $1/(1-\gamma)$ in the first term, we can use the definition of $\mathcal{U}_k$ from Lemma 3.1 to write the resulting minimization problem as

$$\min_{\mathbf{u}} \ \mathbf{u}'\mathbf{y} \quad \text{s.t.} \quad (\mathbf{u} - \hat{\mathbf{g}}_{k,\mathbf{Q}})' \boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1} (\mathbf{u} - \hat{\mathbf{g}}_{k,\mathbf{Q}}) \leq \sigma^2 R_n^2. \tag{A.1}$$

This is the minimization of a linear function subject to a convex quadratic constraint. The Lagrangian corresponding to (A.1) can be written

$$G(\mathbf{u}, \nu) = \mathbf{u}'\mathbf{y} + \nu\left[(\mathbf{u} - \hat{\mathbf{g}}_{k,\mathbf{Q}})'\boldsymbol{\Sigma}_{k,\mathbf{Q}}^{-1}(\mathbf{u} - \hat{\mathbf{g}}_{k,\mathbf{Q}}) - \sigma^2 R_n^2\right],$$

where $\nu \geq 0$ is the Lagrange multiplier associated with the constraint. The dual function is given by $D(\nu) = \min_{\mathbf{u}} G(\mathbf{u}, \nu)$. By applying sufficient conditions to $G(\mathbf{u}, \nu)$, we can write the dual function in closed-form as

$$D(\nu) = \hat{\mathbf{g}}'_{k,\mathbf{Q}}\mathbf{y} - \frac{1}{4\nu}\mathbf{y}'\boldsymbol{\Sigma}_{k,\mathbf{Q}}\mathbf{y} - \nu\sigma^2 R_n^2.$$

The corresponding dual problem is $\max_{\nu \geq 0} D(\nu)$. $D(\nu)$ is concave in ν for $\nu > 0$, so we can apply sufficient conditions to find the solution to the dual problem. The optimal value of the dual problem is given by

$$\max_{\nu \geq 0} D(\nu) = \hat{\mathbf{g}}'_{k,\mathbf{Q}}\mathbf{y} - \sigma R_n \sqrt{\mathbf{y}'\boldsymbol{\Sigma}_{k,\mathbf{Q}}\mathbf{y}}. \tag{A.2}$$

By strong duality, this is the optimal value of the primal problem in (A.1). After rescaling (A.2) by $1/(1-\gamma)$ and including the second term of $f(\mathbf{u})$ that we omitted to begin the proof, we see that $\min_{\mathbf{u} \in \mathcal{U}_k} f(\mathbf{u})$ is equivalent to the right-hand side of (3.9). □

A.2 Detailed Proofs for Chapter 4

A.2.1 Useful Results

We will make use of the following results in our proof of Lemma 4.1.

Lemma A.2 (Kakade and Langford 2002)**.** *Consider any policy π and a current policy π_k. Then,*

$$J(\pi) - J(\pi_k) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s \sim d^{\pi}} \left[\mathop{\mathbb{E}}_{a \sim \pi(\,\cdot\,|s)} \left[A^{\pi_k}(s, a) \right] \right].$$

Lemma A.3 (Achiam et al. 2017)**.** *Consider any policy π and a reference policy π_{ref}. Then,*

$$\text{TV}\left(d^{\pi}, d^{\pi_{\text{ref}}}\right) \leq \frac{\gamma}{1-\gamma} \mathop{\mathbb{E}}_{s \sim d^{\pi_{\text{ref}}}} \left[\text{TV}\left(\pi, \pi_{\text{ref}}\right)(s) \right],$$

where d^{π} and $d^{\pi_{\text{ref}}}$ represent normalized discounted state visitation distributions.

A.2.2 Detailed Proof of Lemma 4.1

Proof. We apply similar proof techniques as in Achiam et al. (2017), but we are interested in $d^{\pi_{\text{ref}}}$ as our sampling distribution rather than d^{π_k}. Starting from the

equality in Lemma A.2, we add and subtract the term

$$\frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right].$$

By doing so, we have

$$\begin{aligned} J(\pi) - J(\pi_k) &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] \\ &\quad + \frac{1}{1-\gamma} \left(\mathop{\mathbb{E}}_{s\sim d^{\pi}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] - \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] \right) \\ &\geq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] \\ &\quad - \frac{1}{1-\gamma} \left| \mathop{\mathbb{E}}_{s\sim d^{\pi}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] - \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] \right|. \end{aligned} \tag{A.3}$$

Next, we can bound the magnitude of the second term in the right-hand side of (A.3). We have that

$$\begin{aligned} &\frac{1}{1-\gamma} \left| \mathop{\mathbb{E}}_{s\sim d^{\pi}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] - \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] \right| \\ &\qquad \leq \frac{1}{1-\gamma} \cdot \max_{s\in\mathcal{S}} \left| \mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right| \cdot \int_{\mathcal{S}} \left| d^{\pi}(s) - d^{\pi_{\text{ref}}}(s) \right| \mathrm{d}s && \text{(A.4)} \\ &\qquad = \frac{2C^{\pi,\pi_k}}{1-\gamma} \cdot \mathrm{TV}\left(d^{\pi}, d^{\pi_{\text{ref}}}\right) && \text{(A.5)} \\ &\qquad \leq \frac{2\gamma C^{\pi,\pi_k}}{(1-\gamma)^2} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathrm{TV}\left(\pi, \pi_{\text{ref}}\right)(s) \right], && \text{(A.6)} \end{aligned}$$

where in (A.4) we applied Hölder's inequality, in (A.5) we used the definition of C^{π,π_k} from Lemma 2.1 and the definition of TV distance, and in (A.6) we applied the result from Lemma A.3. This leads to the lower bound

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\,\cdot\,|s)} \left[A^{\pi_k}(s,a) \right] \right] - \frac{2\gamma C^{\pi,\pi_k}}{(1-\gamma)^2} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\text{ref}}}} \left[\mathrm{TV}\left(\pi, \pi_{\text{ref}}\right)(s) \right].$$

Finally, we apply importance sampling on the actions in the first term to obtain the result, which assumes that the support of π is contained within the support of π_{ref} at every state. □

A.3 Detailed Proofs for Chapter 5

In this section, we prove all results for the robust cost Bellman operator $\mathcal{T}^{\pi}_{\mathcal{P},c}$. Note that

$$\inf_{p_{s,a}\in\mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[V_r^{\pi}(s')\right] = -\sup_{p_{s,a}\in\mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[-V_r^{\pi}(s')\right].$$

Therefore, results related to the robust reward Bellman operator $\mathcal{T}^{\pi}_{\mathcal{P},r}$ follow immediately by applying the same proofs after an appropriate change in signs.

A.3.1 Useful Results

In order to prove the tractable reformulation in Theorem 5.1, we will make use of the following result.

Lemma A.4. *Let Assumption 5.1 hold. Then, we have*

$$\mathcal{T}^{\pi}_{\mathcal{P},r}Q_r(s,a) = r(s,a) + \gamma \sup_{\lambda\geq 0} \mathop{\mathbb{E}}_{\hat{s}'\sim\hat{p}_{s,a}} \left[\inf_{s'\in\mathcal{S}} V_r^{\pi}(s') + \lambda\left(d_{s,a}(\hat{s}',s') - \epsilon_{s,a}\right)\right], \quad \text{(A.7)}$$

$$\mathcal{T}^{\pi}_{\mathcal{P},c}Q_c(s,a) = c(s,a) + \gamma \inf_{\lambda\geq 0} \mathop{\mathbb{E}}_{\hat{s}'\sim\hat{p}_{s,a}} \left[\sup_{s'\in\mathcal{S}} V_c^{\pi}(s') - \lambda\left(d_{s,a}(\hat{s}',s') - \epsilon_{s,a}\right)\right]. \quad \text{(A.8)}$$

Proof. Under Assumption 5.1, note that Assumption (A1) and Assumption (A2) of Blanchet and Murthy (2019) are satisfied for the distributionally robust optimization problem in (5.3). Assumption (A1) is satisfied by our definition of optimal transport cost, and Assumption (A2) is satisfied by our Assumption 5.1. Then, according to Theorem 1 in Blanchet and Murthy (2019), optimal transport strong duality holds for the distributionally robust optimization problem in (5.3). Therefore, we have that

$$\sup_{p_{s,a}\in\mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[V_c^{\pi}(s')\right] = \inf_{\lambda\geq 0} \mathop{\mathbb{E}}_{\hat{s}'\sim\hat{p}_{s,a}} \left[\sup_{s'\in\mathcal{S}} V_c^{\pi}(s') - \lambda\left(d_{s,a}(\hat{s}',s') - \epsilon_{s,a}\right)\right].$$

By substituting this result into (5.3), we arrive at the result in (A.8). □

A.3.2 Detailed Proof of Theorem 5.1

Proof. First, we show that (5.8) and the distributionally robust optimization problem in (5.3) share the same dual problem. We write the dual problem to (5.8) as

$$\inf_{\lambda \geq 0} \sup_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} [V_c^\pi(g(\hat{s}'))] - \lambda \left(\mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} [d_{s,a}(\hat{s}', g(\hat{s}'))] - \epsilon_{s,a} \right)$$
$$= \inf_{\lambda \geq 0} \sup_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[V_c^\pi(g(\hat{s}')) - \lambda \left(d_{s,a}(\hat{s}', g(\hat{s}')) - \epsilon_{s,a} \right) \right].$$

Using the definition of $\mathcal{G}$, we can rewrite this as

$$\inf_{\lambda \geq 0} \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[\sup_{s' \in \mathcal{S}} V_c^\pi(s') - \lambda \left(d_{s,a}(\hat{s}', s') - \epsilon_{s,a} \right) \right], \tag{A.9}$$

which appears in the right-hand side of (A.8) from Lemma A.4. Note that (A.9) is also the dual to the distributionally robust optimization problem in (5.3) and optimal transport strong duality holds (see Lemma A.4).

Next, we show that strong duality holds between (5.8) and (A.9). Let λ^* be the optimal dual variable in (A.9), and let

$$g_{s,a}^*(\hat{s}') \in \arg\max_{s' \in \mathcal{S}} V_c^\pi(s') - \lambda^* \left(d_{s,a}(\hat{s}', s') - \epsilon_{s,a} \right).$$

We have that λ^* and $g_{s,a}^*(\hat{s}')$ exist according to Theorem 1(b) in Blanchet and Murthy (2019) along with Assumption 5.2, and $g_{s,a}^*$ characterizes the optimal transport plan ν^* that moves the probability of $\hat{s}'$ under $\hat{p}_{s,a}$ to $g_{s,a}^*(\hat{s}')$. By the complementary slackness results of Theorem 1(b) in Blanchet and Murthy (2019), we also have that

$$\lambda^* \left(\mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[d_{s,a}(\hat{s}', g_{s,a}^*(\hat{s}')) \right] - \epsilon_{s,a} \right) = 0.$$

Therefore,

$$\inf_{\lambda \geq 0} \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[\sup_{s' \in \mathcal{S}} V_c^\pi(s') - \lambda \left(d_{s,a}(\hat{s}', s') - \epsilon_{s,a} \right) \right]$$
$$= \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[V_c^\pi(g_{s,a}^*(\hat{s}')) - \lambda^* \left(d_{s,a}(\hat{s}', g_{s,a}^*(\hat{s}')) - \epsilon_{s,a} \right) \right]$$
$$= \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[V_c^\pi(g_{s,a}^*(\hat{s}')) \right].$$

Moreover, by the primal feasibility of the optimal transport plan ν^*, we have that

$$\mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[d(\hat{s}', g^*_{s,a}(\hat{s}')) \right] \leq \epsilon_{s,a},$$

so $g^*_{s,a}$ is a feasible solution to (5.8) with the same objective value as the value of (A.9). Therefore, strong duality holds between (5.8) and (A.9), and $g^*_{s,a}$ is an optimal solution to (5.8) (i.e., an optimal solution to (5.8) exists). Then, for any optimal solution $g^c_{s,a}$ to (5.8), we have that

$$\mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[V^\pi_c(g^c_{s,a}(\hat{s}')) \right] = \mathop{\mathbb{E}}_{\hat{s}' \sim \hat{p}_{s,a}} \left[V^\pi_c(g^*_{s,a}(\hat{s}')) \right],$$

and the right-hand side of (A.8) is equivalent to the right-hand side of (5.6). □

A.4 Detailed Proofs for Chapter 6

In this section, we prove all results related to the RAMU cost Bellman operator $\mathcal{T}^\pi_{\rho,c}$. Using the fact that $\rho^+(Z) = -\rho(-Z)$ for a coherent distortion risk measure ρ on a cost random variable, all results related to the RAMU reward Bellman operator follow by an appropriate change in sign.

A.4.1 Useful Results

Consider the probability space $(\mathcal{M}, \mathcal{F}, \mu_{s,a})$, where $\mathcal{F}$ is a σ-algebra on $\mathcal{M}$ and $\mu_{s,a} \in P(\mathcal{M})$ defines a probability measure over $\mathcal{M}$. Let $\mathcal{Z}$ be a space of random variables defined on this probability space, and let $\mathcal{Z}^*$ be its corresponding dual space. In order to prove Theorem 6.1, we make use of the following dual representation of coherent risk measures applied to our probability space of interest.

Lemma A.5 (Shapiro et al. 2014)**.** *Let ρ be a proper, real-valued coherent risk measure. Then, for any $Z \in \mathcal{Z}$ we have that*

$$\rho(Z) = \sup_{\beta_{s,a} \in \mathcal{U}_{s,a}} \mathbb{E}_{\beta_{s,a}}[Z],$$

where $\mathbb{E}_{\beta_{s,a}}[\cdot]$ represents expectation with respect to the probability measure $\beta_{s,a} \in$

$P(\mathcal{M})$, and

$$\mathcal{U}_{s,a} \subseteq \{\beta_{s,a} \in P(\mathcal{M}) \mid \beta_{s,a} = \xi_{s,a}\mu_{s,a},\ \xi_{s,a} \in \mathcal{Z}^*\}$$

is a convex, bounded, and weakly closed set that depends on ρ.*

See Shapiro et al. (2014) for a general treatment of this result.

A.4.2 Detailed Proof of Theorem 6.1

Proof. For a given state-action pair, we apply Lemma A.5 to the risk measure that appears in the formulation of $\mathcal{T}^{\pi}_{\rho,c}$ given by Lemma 6.1. By doing so, we have that

$$\begin{aligned}
\mathcal{T}^{\pi}_{\rho,c}Q(s,a) &= \underset{p_{s,a}\sim\mu_{s,a}}{\rho}\left(\mathcal{T}^{\pi}_{p,c}Q(s,a)\right) \\
&= \sup_{\beta_{s,a}\in\mathcal{U}_{s,a}} \underset{p_{s,a}\sim\beta_{s,a}}{\mathbb{E}}\left[\mathcal{T}^{\pi}_{p,c}Q(s,a)\right] \\
&= c(s,a) + \gamma \sup_{\beta_{s,a}\in\mathcal{U}_{s,a}} \underset{p_{s,a}\sim\beta_{s,a}}{\mathbb{E}}\left[\underset{s'\sim p_{s,a}}{\mathbb{E}}\left[\underset{a'\sim\pi(\cdot|s')}{\mathbb{E}}\left[Q(s',a')\right]\right]\right],
\end{aligned}$$

where $\mathcal{U}_{s,a}$ is defined in Lemma A.5. Therefore, $\mathcal{T}^{\pi}_{\rho,c}$ has the same form as a distributionally robust Bellman operator (Xu and Mannor, 2010; Yu and Xu, 2016) with the ambiguity set $\mathcal{U} = \bigotimes_{(s,a)\in\mathcal{S}\times\mathcal{A}}\mathcal{U}_{s,a}$. □

A.4.3 Detailed Proof of Corollary 6.1

Given the equivalence to a distributionally robust Bellman operator as shown in Theorem 6.1, Corollary 6.1 follows from results in Xu and Mannor (2010) and Yu and Xu (2016). We include a proof for completeness.

Proof. Due to the linearity of the expectation operator, for a given $\beta_{s,a} \in \mathcal{U}_{s,a}$ we have that

$$\underset{p_{s,a}\sim\beta_{s,a}}{\mathbb{E}}\left[\underset{s'\sim p_{s,a}}{\mathbb{E}}\left[\underset{a'\sim\pi(\cdot|s')}{\mathbb{E}}\left[Q(s',a')\right]\right]\right] = \underset{s'\sim\bar{p}^{\beta}_{s,a}}{\mathbb{E}}\left[\underset{a'\sim\pi(\cdot|s')}{\mathbb{E}}\left[Q(s',a')\right]\right],$$

where $\bar{p}^{\beta}_{s,a} = \mathbb{E}_{p_{s,a}\sim\beta_{s,a}}\left[p_{s,a}\right] \in P(\mathcal{S})$ represents a mixture transition model determined

by $\beta_{s,a}$. Therefore, starting from the result in Theorem 6.1, we can write

$$\begin{aligned}\mathcal{T}^{\pi}_{\rho,c}Q(s,a) &= c(s,a) + \gamma \sup_{\beta_{s,a}\in\mathcal{U}_{s,a}} \mathop{\mathbb{E}}_{p_{s,a}\sim\beta_{s,a}} \left[\mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a'\sim\pi(\cdot|s')} [Q(s',a')] \right] \right] \\ &= c(s,a) + \gamma \sup_{\bar{p}^{\beta}_{s,a}\in\mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s'\sim\bar{p}^{\beta}_{s,a}} \left[\mathop{\mathbb{E}}_{a'\sim\pi(\cdot|s')} [Q(s',a')] \right],\end{aligned}$$

where

$$\mathcal{P}_{s,a} = \left\{ \bar{p}^{\beta}_{s,a} \in P(\mathcal{S}) \mid \bar{p}^{\beta}_{s,a} = \mathop{\mathbb{E}}_{p_{s,a}\sim\beta_{s,a}} [p_{s,a}],\ \beta_{s,a} \in \mathcal{U}_{s,a} \right\}.$$

As a result, $\mathcal{T}^{\pi}_{\rho,c}$ has the same form as a robust Bellman operator (Iyengar, 2005; Nilim and Ghaoui, 2005) with the uncertainty set $\mathcal{P} = \bigotimes_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{P}_{s,a}$. Similarly, the RAMU reward Bellman operator $\mathcal{T}^{\pi}_{\rho^+,r}$ has the same form as a robust Bellman operator with the uncertainty set $\mathcal{P}^+ = \bigotimes_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{P}^+_{s,a}$, where

$$\mathcal{P}^+_{s,a} = \left\{ \bar{p}^{\beta}_{s,a} \in P(\mathcal{S}) \mid \bar{p}^{\beta}_{s,a} = \mathop{\mathbb{E}}_{p_{s,a}\sim\beta_{s,a}} [p_{s,a}],\ \beta_{s,a} \in \mathcal{U}^+_{s,a} \right\}.$$

□

A.4.4 Detailed Proof of Corollary 6.2

Corollary 6.2 also follows from previous results on distributionally robust Bellman operators (Xu and Mannor, 2010; Yu and Xu, 2016) and robust Bellman operators (Iyengar, 2005; Nilim and Ghaoui, 2005) due to the equivalences shown in Theorem 6.1 and Corollary 6.1. Again, we include a proof for completeness.

Proof. From Corollary 6.1, we can write

$$\mathcal{T}^{\pi}_{\rho,c}Q(s,a) = c(s,a) + \gamma \sup_{p_{s,a}\in\mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s'\sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a'\sim\pi(\cdot|s')} [Q(s',a')] \right].$$

Consider Q functions $Q^{(1)}$ and $Q^{(2)}$, and denote the sup-norm by

$$\|Q^{(1)} - Q^{(2)}\|_{\infty} = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left| Q^{(1)}(s,a) - Q^{(2)}(s,a) \right|.$$

Fix $\epsilon > 0$ and consider $(s, a) \in \mathcal{S} \times \mathcal{A}$. Then, there exists $p_{s,a}^{(1)} \in \mathcal{P}_{s,a}$ such that

$$\mathop{\mathbb{E}}_{s' \sim p_{s,a}^{(1)}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \left[Q^{(1)}(s', a') \right] \right] \geq \sup_{p_{s,a} \in \mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s' \sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \left[Q^{(1)}(s', a') \right] \right] - \epsilon.$$

We have that

$$\begin{aligned}
&\mathcal{T}_{\rho,c}^{\pi} Q^{(1)}(s, a) - \mathcal{T}_{\rho,c}^{\pi} Q^{(2)}(s, a) \\
&= \gamma \left(\sup_{p_{s,a} \in \mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s' \sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \left[Q^{(1)}(s', a') \right] \right] - \sup_{p_{s,a} \in \mathcal{P}_{s,a}} \mathop{\mathbb{E}}_{s' \sim p_{s,a}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \left[Q^{(2)}(s', a') \right] \right] \right) \\
&\leq \gamma \left(\mathop{\mathbb{E}}_{s' \sim p_{s,a}^{(1)}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \left[Q^{(1)}(s', a') \right] \right] + \epsilon - \mathop{\mathbb{E}}_{s' \sim p_{s,a}^{(1)}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \left[Q^{(2)}(s', a') \right] \right] \right) \\
&= \gamma \mathop{\mathbb{E}}_{s' \sim p_{s,a}^{(1)}} \left[\mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \left[Q^{(1)}(s', a') - Q^{(2)}(s', a') \right] \right] + \gamma\epsilon \\
&\leq \gamma \| Q^{(1)} - Q^{(2)} \|_{\infty} + \gamma\epsilon.
\end{aligned}$$

A similar argument can be used to show that

$$-\gamma \| Q^{(1)} - Q^{(2)} \|_{\infty} - \gamma\epsilon \leq \mathcal{T}_{\rho,c}^{\pi} Q^{(1)}(s, a) - \mathcal{T}_{\rho,c}^{\pi} Q^{(2)}(s, a),$$

so we have that

$$\left| \mathcal{T}_{\rho,c}^{\pi} Q^{(1)}(s, a) - \mathcal{T}_{\rho,c}^{\pi} Q^{(2)}(s, a) \right| \leq \gamma \| Q^{(1)} - Q^{(2)} \|_{\infty} + \gamma\epsilon.$$

By applying a supremum over state-action pairs on the left-hand side, we obtain

$$\| \mathcal{T}_{\rho,c}^{\pi} Q^{(1)} - \mathcal{T}_{\rho,c}^{\pi} Q^{(2)} \|_{\infty} \leq \gamma \| Q^{(1)} - Q^{(2)} \|_{\infty} + \gamma\epsilon.$$

Finally, since $\epsilon > 0$ was arbitrary, we have shown that $\mathcal{T}_{\rho,c}^{\pi}$ is a γ-contraction in the sup-norm. $\square$

Appendix B

Implementation Details

B.1 Implementation Details for Chapter 3 and Chapter 4

B.1.1 Advantage Estimation

For all of the policy improvement algorithms we consider in Chapter 3 and Chapter 4, we must estimate the advantage function $A^{\pi_k}(s, a)$ of the current policy π_k. The use of $A^{\pi_k}(s, a)$ is important because it allows our methods to provide policy improvement guarantees with respect to the current policy, regardless of the policy used to generate the data. In the on-policy setting, advantage estimation is straightforward because multi-step advantage estimates are unbiased except for the use of bootstrapping with the learned value function. We use Generalized Advantage Estimation (GAE) (Schulman et al., 2016) with $\lambda = 0.97$ in all on-policy algorithms (including UA-TRPO in Chapter 3), where λ determines a weighted average over K-step advantage estimates.

When estimating $A^{\pi_k}(s, a)$ for our GPI algorithms in Chapter 4, on the other hand, our data has been collected using prior policies. As a result, the multi-step estimates used in GAE are no longer unbiased. Therefore, we must use off-policy estimation techniques to account for this bias. We consider an off-policy variant of GAE that uses the V-trace value function estimator (Espeholt et al., 2018), which corrects multi-step off-policy estimates while controlling variance via truncated importance sampling. For data collected under a prior policy π_{k-i}, the K-step V-trace estimate of the current

value function is given by

$$V^{\pi_k}_{\text{trace}}(s_t) = V(s_t) + \sum_{j=0}^{K-1} \gamma^j \left(\prod_{m=0}^{j} c_{t+m} \right) \delta^V_{t+j},$$

where V is a learned value function, $\delta^V_t = r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$, and $c_t = \min(\bar{c}, {}^{\pi_k(a_t|s_t)}/_{\pi_{k-i}(a_t|s_t)})$ represents a truncated importance sampling ratio with truncation parameter $\bar{c}$. Similarly, for $K \geq 2$, the corrected K-step estimate of the current advantage function using V-trace is given by

$$A^{\pi_k}_{\text{trace}}(s_t, a_t) = \delta^V_t + \sum_{j=1}^{K-1} \gamma^j \left(\prod_{m=1}^{j} c_{t+m} \right) \delta^V_{t+j},$$

and for $K = 1$ we have the standard one-step estimate $A^{\pi_k}_{\text{trace}}(s_t, a_t) = \delta^V_t$ that does not require any correction. Note that Espeholt et al. (2018) treat the final importance sampling ratio in each term separately, but we do not make this distinction in our notation because the truncation parameter is typically chosen to be the same for all terms. We use the default setting of $\bar{c} = 1$ from Espeholt et al. (2018) in our experiments, and as in GAE we consider a weighted average over K-step estimates determined by the parameter λ.

It is common to standardize advantage estimates within each batch or minibatch during policy updates. Note that the expectation of $A^{\pi_k}(s, a)$ with respect to samples generated under the current policy π_k equals zero, so standardization ensures that our sample-based estimates also satisfy this property. Therefore, the appropriate quantity to standardize in the generalized case is

$$\frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s, a),$$

since the expectation of this term with respect to data generated under prior policies equals zero.

Table B.1: Network architectures and hyperparameter values shared across experiments in Chapter 3 and Chapter 4.

General	
Discount rate (γ)	0.995
Trust region parameter (ϵ)	0.2
Policy	
Layer sizes	64, 64
Layer activations	tanh
Initial standard deviation	1.0
Value Function	
Layer sizes	64, 64
Layer activations	tanh
Optimizer	Adam
Learning rate	3e$-$4
Minibatches per epoch	32
Epochs per update	10
GAE parameter (λ)	0.97
Chapter 3	
Batch size (N)	1,024
Chapter 4	
On-policy batch size (N)	2,048
Minimum batch size (n)	1,024
Trade-off parameter (κ)	$0.0, 0.5, 1.0$
V-trace truncation parameter ($\bar{c}$)	1.0

B.1.2 Network Architectures

As discussed in Chapter 3 and Chapter 4, we represent the policy π as a multivariate Gaussian distribution where the mean action for a given state is parameterized by a neural network with two hidden layers of 64 units each and tanh activations. The state-independent standard deviation is parameterized separately. The value function is parameterized by a separate neural network with two hidden layers of 64 units each and tanh activations, and is updated at every iteration using minibatch stochastic gradient descent.

Table B.2: Hyperparameter values by algorithm for experiments in Chapter 3 and Chapter 4.

UA-TRPO	
Trade-off parameter (c)	0.1
Confidence parameter (α)	0.05
Number of random projections (m)	200
Number of minibatch gradient estimates	256
PPO	
Policy optimizer	Adam
Initial policy learning rate (η)	3e−4
Adaptive learning rate factor (υ)	0.03
Policy minibatches per epoch	32
Policy epochs per update	10
TRPO / VMPO	
Conjugate gradient iterations per update	20
Conjugate gradient damping coefficient	0.01

B.1.3 Algorithm Hyperparameters

For our experiments in Chapter 3 and Chapter 4, we provide the values of all hyperparameters in Table B.1 and Table B.2. In addition, we describe how to calculate the scaling coefficients $c_{\text{ESS}}, c_{\text{TV}}$ used when determining the optimal mixture distribution for sample reuse in Theorem 4.3. We include these coefficients in Theorem 4.3 so that each component of the objective is on the same scale, which we can accomplish by setting each coefficient to be the range of potential values for its corresponding numerator. The numerator in the effective sample size term is the largest when $\kappa = 0$ and smallest when $\kappa = 1$, and the reverse is true for the numerator in the total TV distance update size term. Therefore, we set the scaling coefficients to be

$$c_{\text{ESS}} = \sum_i \nu_i^*(0)^2 - \sum_i \nu_i^*(1)^2, \qquad c_{\text{TV}} = \sum_i \nu_i^*(1)(i+1) - \sum_i \nu_i^*(0)(i+1),$$

where $\nu^*(0)$ and $\nu^*(1)$ are the optimal mixture distributions when $\kappa = 0$ and $\kappa = 1$, respectively. Note that we can calculate these optimal mixture distributions by

Table B.3: Safety constraints for all tasks.

Task	Safety Constraint	Safety Coefficient
Cartpole Swingup	Slider Position	0.30
Walker Walk	Joint Velocity	0.25
Walker Run	Joint Velocity	0.30
Quadruped Walk	Joint Angle	0.15
Quadruped Run	Joint Angle	0.30

ignoring the scaling coefficients in Theorem 4.3, since the coefficients do not impact the resulting minimizer for these extreme values of trade-off parameter κ.

B.2 Implementation Details for Chapter 5 and Chapter 6

B.2.1 Safety Constraints and Environment Perturbations

In Chapter 5 and Chapter 6, we consider experiments on tasks from the Real-World RL Suite where the goal is to optimize a task objective while satisfying a safety constraint. For each task, we apply a single safety constraint corresponding to a cost function defined in the Real-World RL Suite, which we summarize in Table B.3. See Dulac-Arnold et al. (2021) for detailed definitions of each safety constraint.

The definitions of these cost functions depend on a safety coefficient in $[0, 1]$. As the safety coefficient decreases, the range of safe outcomes also decreases and the safety constraints corresponding to these cost functions become more difficult to satisfy. In order to consider safe RL tasks with difficult safety constraints where strong performance is still possible, we selected the value of this safety coefficient in the range of $[0.15, 0.20, 0.25, 0.30]$ for each task based on the performance of the baseline safe RL algorithm CRPO compared to the unconstrained algorithm MPO. Figure B·1 shows total rewards throughout training for each task across this range of safety coefficients. We selected the most difficult cost definition in this range (i.e., lowest safety coefficient value) where CRPO is still able to achieve the same total

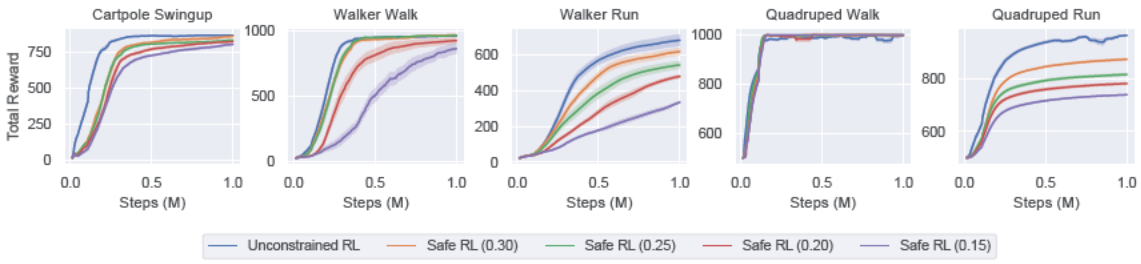

Figure B·1: Hyperparameter sweep of safety coefficient. Value in parentheses represents safety coefficient used for training in safe RL. Shading denotes half of one standard error across policies.

rewards as MPO (or the value that leads to the smallest gap between the two in the case of Walker Run and Quadruped Run). The resulting safety coefficients used for our experiments are listed in Table B.3.

In order to evaluate the robustness of our learned policies, we generate a range of test environments for each task based on perturbing a simulator parameter in the Real-World RL Suite. See Table B.4 for the perturbation parameters and corresponding ranges considered in our experiments. Note that the parameter value associated with the nominal training environment is in the center of the range of parameter values considered at test time.

Finally, note that the domain randomization baselines consider a range of environments during training. In-distribution domain randomization applies a uniform distribution over the middle 50% of the parameter values considered at test time. The out-of-distribution variant of domain randomization instead considers a uniform distribution over a range of values for a different simulator parameter than the one varied at test time. See Table B.5 for details.

B.2.2 Network Architectures

For our experiments in Chapter 5 and Chapter 6, we consider neural network representations of the policy and critics. We consider networks with 3 hidden layers of 256

Table B.4: Perturbation ranges for test environments.

Domain	Perturbation Parameter	Nominal Value	Test Range
Cartpole	Pole Length	1.00	$[0.75, 1.25]$
Walker	Torso Length	0.30	$[0.10, 0.50]$
Quadruped	Torso Density	1,000	$[500, 1{,}500]$

Table B.5: Perturbation ranges for domain randomization.

Domain	Perturbation Parameter	Nominal Value	Training Range
In-Distribution			
Cartpole	Pole Length	1.00	$[0.875, 1.125]$
Walker	Torso Length	0.30	$[0.20, 0.40]$
Quadruped	Torso Density	1,000	$[750, 1{,}250]$
Out-of-Distribution			
Cartpole	Pole Mass	0.10	$[0.05, 0.15]$
Walker	Contact Friction	0.70	$[0.40, 1.00]$
Quadruped	Contact Friction	1.50	$[1.00, 2.00]$

units and ELU activations, and we apply layer normalization followed by a tanh activation after the first layer as in Abdolmaleki et al. (2020). We represent the policy as a multivariate Gaussian distribution with diagonal covariance, where at a given state the policy network outputs the mean $\mu(s)$ and diagonal covariance $\Sigma(s)$ of the action distribution. The diagonal of $\Sigma(s)$ is calculated by applying the softplus operator to the outputs of the neural network corresponding to the covariance. In addition to the policy network, we consider separate networks for the reward and cost critics. We maintain target versions of the policy and critic networks using an exponential moving average of the weights with $\tau = 5\text{e}{-3}$. Finally, for our OTP framework in Chapter 5, we also consider neural networks for our perturbation networks δ_r and δ_c. We consider small networks with 2 hidden layers of 64 units and ELU activations, and we clip the outputs in the range $[-2\epsilon_\delta, 2\epsilon_\delta]$ for additional stability.

Table B.6: Network architectures and hyperparameter values shared across experiments in Chapter 5 and Chapter 6.

General	
Batch size per update	256
Updates per environment step	1
Discount rate (γ)	0.99
Target network exponential moving average (τ)	5e−3
Policy	
Layer sizes	256, 256, 256
Layer activations	ELU
Layer norm + tanh on first layer	Yes
Initial standard deviation	0.3
Optimizer	Adam
Learning rate	1e−4
Non-parametric KL (ϵ_{KL})	0.10
Action penalty KL	1e−3
Action samples per update	20
Parametric mean KL (β_μ)	0.01
Parametric covariance KL (β_Σ)	1e−5
Parametric KL dual learning rate	0.01
Critics	
Layer sizes	256, 256, 256
Layer activations	ELU
Layer norm + tanh on first layer	Yes
Optimizer	Adam
Learning rate	1e−4

B.2.3 Algorithm Hyperparameters

We consider CRPO (Xu et al., 2021) as the baseline safe RL algorithm for all of our experiments in Chapter 5 and Chapter 6. At every update, CRPO calculates the current value of the safety constraint based on a batch of sampled data. If the safety constraint is satisfied for the current batch, it applies a policy update to maximize rewards. Otherwise, it applies a policy update to minimize costs. In both cases, we use the unconstrained RL algorithm MPO (Abdolmaleki et al., 2018) to calculate policy updates. MPO calculates a non-parametric target policy with KL divergence ϵ_{KL} from the current policy, and updates the current policy towards this target while constraining separate KL divergence contributions from the mean and covariance by β_μ

and β_Σ, respectively. We apply per-dimension KL divergence constraints and action penalization using the multi-objective MPO framework (Abdolmaleki et al., 2020) as in Hoffman et al. (2020), and we consider closed-form updates of the temperature parameter used in the non-parametric target policy as in Liu et al. (2022) to account for the immediate switching between objectives in CRPO. See Table B.6 for all important hyperparameter values associated with the implementation of policy updates using MPO, and see Abdolmaleki et al. (2018) for additional details.

For our OTP framework in Chapter 5, we update the perturbation networks alongside the policy and critics. We combine the perturbation function updates in (5.7) and (5.8) across state-action pairs by averaging over samples from the replay buffer, and we apply a radius of ϵ_δ^2. This leads to updates of the form

$$g_{\delta_r} \in \arg\min_{g_\delta \in \mathcal{F}_\delta} \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} [V_r^\pi(g_\delta(s,a,\hat{s}'))] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} [d_{s,a}(\hat{s}', g_\delta(s,a,\hat{s}'))] \leq \epsilon_\delta^2,$$

$$g_{\delta_c} \in \arg\max_{g_\delta \in \mathcal{F}_\delta} \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} [V_c^\pi(g_\delta(s,a,\hat{s}'))] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{(s,a,\hat{s}')\sim\mathcal{D}} [d_{s,a}(\hat{s}', g_\delta(s,a,\hat{s}'))] \leq \epsilon_\delta^2,$$

where $\mathcal{F}_\delta$ represents the class of perturbation functions with the form in (5.9). We consider $d_{s,a}$ given by (5.4) to arrive at the perturbation network updates in (5.10) and (5.11), where ϵ_δ determines the average per-coordinate magnitude of the outputs of δ_r and δ_c. We consider $\epsilon_\delta = 0.02$ in our experiments. We apply gradient-based updates on the Lagrangian relaxations of (5.10) and (5.11), and we also update the corresponding dual variables throughout training. See Table B.7 for hyperparameter values associated with our OTP framework.

For our RAMU framework in Chapter 6, the latent variable hyperparameter ϵ controls the definition of the distribution $\mu_{s,a}$ over transition models. A larger value of ϵ leads to a distribution over a wider range of transition models, which results in a more robust approach when combined with a risk-averse perspective on model uncertainty. We consider $\epsilon = 0.10$ in our experiments, as it achieves strong constraint

Table B.7: Network architectures and hyperparameter values for OTP and RAMU frameworks.

OTP	
Layer sizes	64, 64
Layer activations	ELU
Layer norm + tanh on first layer	No
Output clipping	$[-2\epsilon_\delta, 2\epsilon_\delta]$
Optimizer	Adam
Learning rate	1e−4
Dual learning rate	0.01
Per-coordinate perturbation magnitude (ϵ_δ)	0.02
RAMU	
Transition model samples per data point (n)	5
Latent variable hyperparameter (ϵ)	0.10

satisfaction without a meaningful decrease in rewards. For computational efficiency we consider $n = 5$ samples of transition models per data point to calculate sample-based Bellman targets in our RAMU framework, as we did not observe meaningful improvements in performance from considering a larger number of samples.

Finally, we also implement adversarial RL and domain randomization using CRPO with MPO policy updates. We represent the adversarial policy in the PR-MDP framework using the same structure and neural network architecture as our main policy, and we train the adversarial policy to maximize costs using MPO. Using the default settings from Tessler et al. (2019a), we apply one adversary update for every 10 policy updates. Domain randomization considers the same updates as the CRPO baseline, but collects data from the range of training environments summarized in Table B.5.

Appendix C

Detailed Experimental Results

C.1 Detailed Results for Chapter 3

In this section, we include detailed results for all experiments considered in Chapter 3. We provide the dimensions of the state and action space for each OpenAI Gym MuJoCo task in Table C.1. In Figure C·1, we include full training curves for all six of these tasks across all three levels of adversarial gradient noise (no noise, 0.5x standard error, and 1.0x standard error). We see that UA-TRPO leads to consistent, robust policy improvement throughout training, even in the presence of adversarial noise. TRPO, on the other hand, struggles to learn under adversarial noise, and leads to performance that is worse than the initial policy in some cases.

C.2 Detailed Results for Chapter 4

Next, we provide detailed experimental results for Chapter 4. We include full training curves in Figure C·2 for the best performing GPI algorithm and the best performing on-policy algorithm across all tasks where learning occurs. In addition, we include details on final performance across all algorithms and all tasks. Figure C·3 shows a comparison of final performance between generalized and on-policy methods for each choice of policy improvement algorithm. We see similar trends across all algorithms, where our generalized approach results in performance improvement across the majority of tasks in the DeepMind Control Suite benchmarking set. The final

Table C.1: Dimensionality of OpenAI Gym MuJoCo tasks.

Task	$\dim(\mathcal{S})$	$\dim(\mathcal{A})$
Swimmer-v3	8	2
Hopper-v3	11	3
HalfCheetah-v3	17	6
Walker2d-v3	17	6
Ant-v3	111	8
Humanoid-v3	376	17

performance of every algorithm across each task is included in Table C.2, as well as the performance of a random Gaussian policy with zero mean and unit standard deviation in each action dimension. We say that learning occurs for a task if the best performing algorithm exceeds the performance of the random policy by at least 10. Finally, we include the sparsity metric associated with each task in Figure C·4.

C.3 Detailed Results for Chapter 5 and Chapter 6

Finally, we include detailed results across tasks and test environments for all baseline algorithms considered in Chapter 5 and Chapter 6. Figure C·5 compares safe RL to both variations of adversarial RL using different probabilities of adversarial intervention. We see that both versions of adversarial RL lead to robust safety in some cases, such as the two Quadruped tasks. In the more adversarial 10% implementation, safety often comes at the cost of overly conservative performance. Adversarial RL with 5% intervention probability achieves stronger total rewards in general, but safety constraint satisfaction in tasks such as Cartpole Swingup is not as robust.

Figure C·6 shows the performance of domain randomization across tasks and test environments. The grey shaded areas represent the training distribution ranges for the in-distribution implementation of domain randomization. We see that domain randomization leads to strong, robust performance in terms of rewards across all test cases, as well as improved constraint satisfaction in perturbed environments compared

to standard safe RL which only considers a single training environment. However, in tasks such as Walker Run and Quadruped Run, domain randomization does not robustly satisfy safety constraints for test environments that were not seen during training. This issue is amplified in the case of out-of-distribution domain randomization, which does not demonstrate consistent robustness benefits compared to standard safe RL. In fact, it even leads to an increase in constraint-violating test cases in Cartpole Swingup compared to safe RL. This demonstrates that training on multiple environments does not necessarily lead to robust performance. Instead, domain knowledge is critical in order for domain randomization to work well in practice.

Table C.2: Final performance across all algorithms and tasks.

		PPO		TRPO		VMPO	
Task	Random	On	GPI	On	GPI	On	GPI
acrobot, swingup	3	**34**	32	21	24	16	24
acrobot, swingup sparse*	0	1	1	1	1	0	1
ball in cup, catch	55	835	879	885	922	884	**939**
cartpole, balance	322	761	**933**	320	624	385	631
cartpole, balance sparse	43	998	**998**	69	410	280	279
cartpole, swingup	47	641	**862**	574	733	620	712
cartpole, swingup sparse	0	166	339	56	285	70	**409**
cheetah, run	5	452	**639**	629	615	594	524
finger, spin	3	494	757	316	**772**	343	729
finger, turn easy	194	187	**243**	176	192	197	211
finger, turn hard	88	83	**127**	82	99	95	96
fish, upright	299	756	**758**	716	754	732	734
fish, swim	65	100	**122**	93	101	104	99
hopper, stand	1	3	**140**	19	50	30	92
hopper, hop	0	22	21	2	13	4	**25**
humanoid, stand*	5	6	7	6	7	6	6
humanoid, walk*	1	2	2	2	2	2	2
humanoid, run*	1	1	1	1	1	1	1
manipulator, bring ball*	0	1	0	1	0	0	1
pendulum, swingup	0	73	199	160	346	82	**490**
point mass, easy	4	**866**	852	141	585	12	406
reacher, easy	50	572	**639**	379	596	309	580
reacher, hard	8	408	**523**	104	420	116	365
swimmer, swimmer6*	192	162	192	154	174	157	178
swimmer, swimmer15*	178	146	157	145	153	146	154
walker, stand	138	516	607	706	**835**	759	804
walker, walk	40	426	590	581	**624**	541	620
walker, run	25	115	191	254	**268**	258	267

Bold indicates best performing algorithm for each task where learning occurs.
Asterisk (*) indicates no learning occurs under any algorithm.

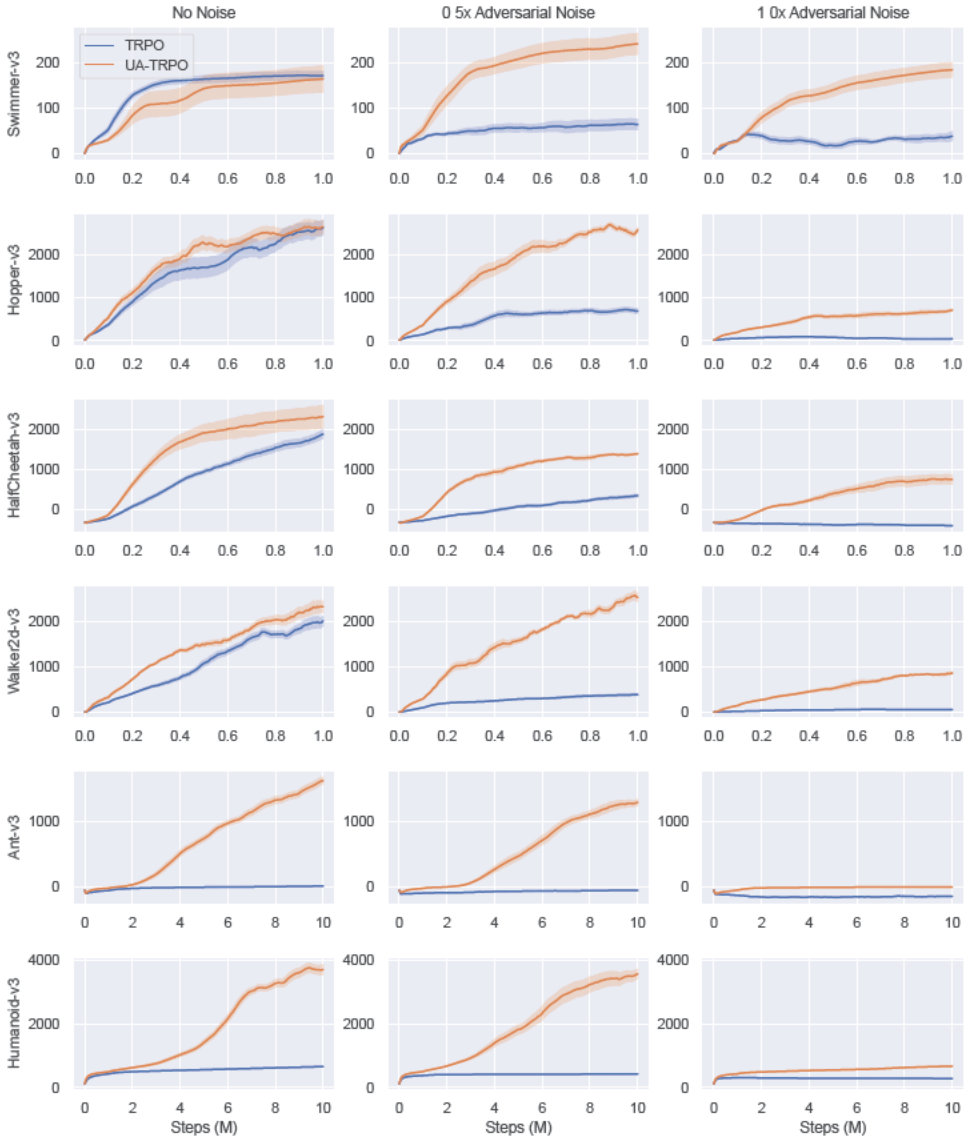

Figure C·1: Average performance of TRPO and UA-TRPO throughout training with different levels of adversarial gradient noise. Shading denotes half of one standard error. Magnitude of adversarial noise is calculated as a multiple of standard error in every dimension of the policy gradient. Sorted left to right by magnitude of adversarial noise. Tasks sorted top to bottom from lowest to highest dimensionality.

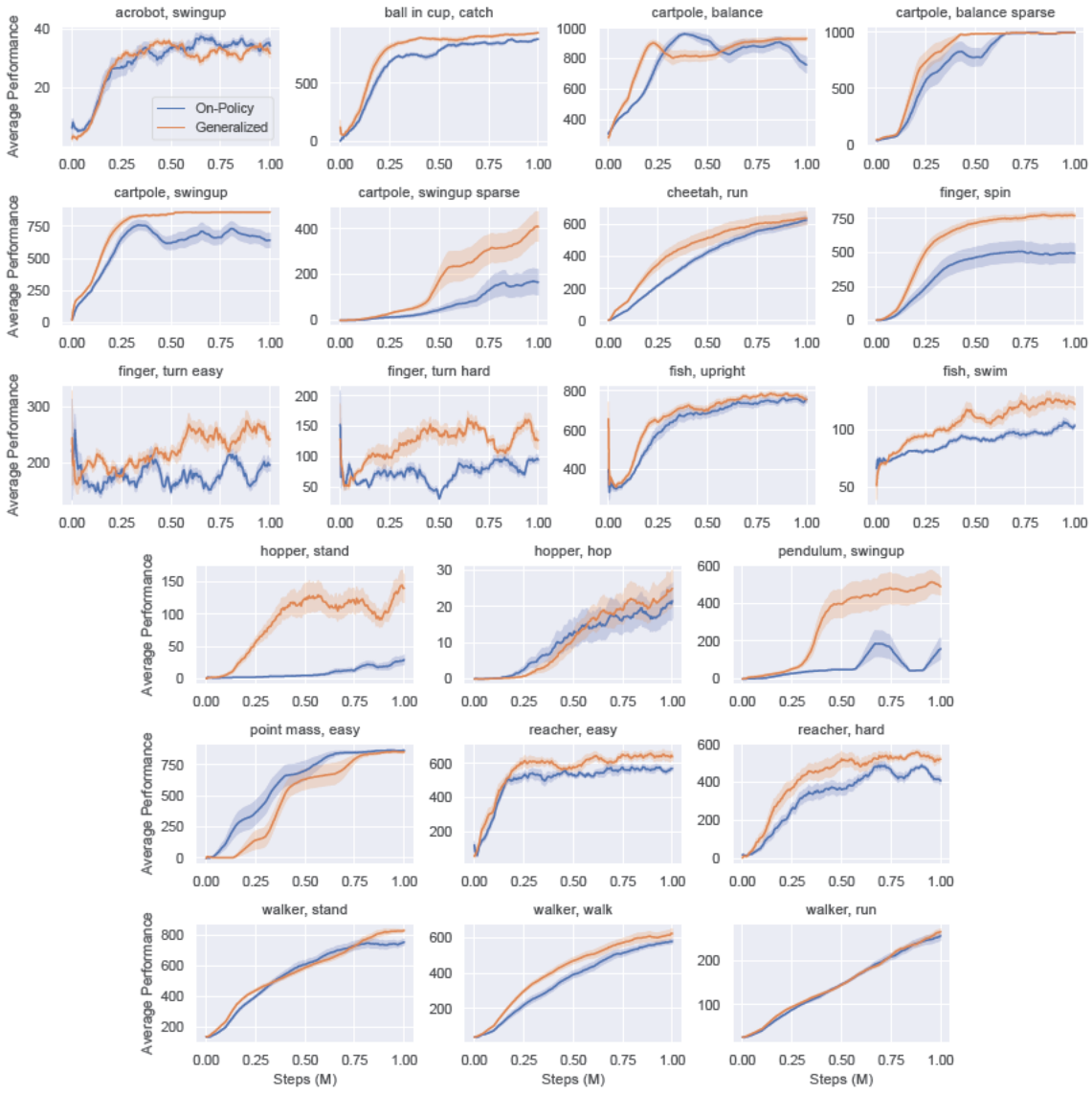

Figure C·2: Generalized vs. on-policy performance throughout training by task. Training curves represent performance of the best performing GPI algorithm and the best performing on-policy algorithm. Shading denotes half of one standard error. Excludes 7 tasks where no learning occurs under any algorithm.

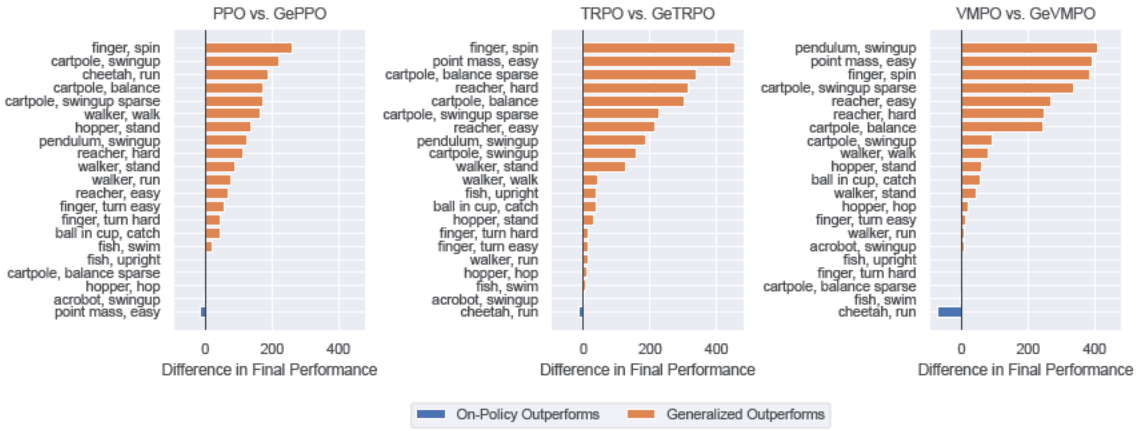

Figure C·3: Difference between generalized and on-policy final performance by algorithm and by task. Bars represent difference in final performance between the GPI algorithm and the on-policy algorithm. Excludes 7 tasks where no learning occurs under any algorithm. Sorted from high to low within each algorithm comparison.

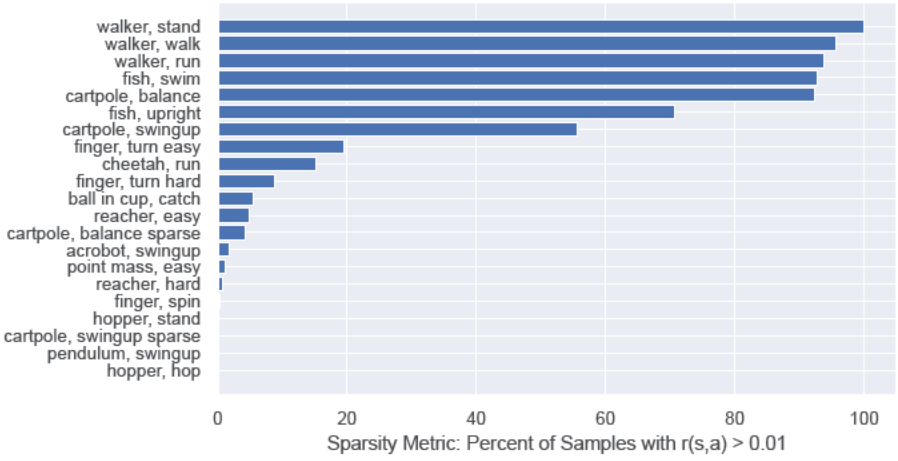

Figure C·4: Sparsity metric by task. Bars represent percent of samples under a random policy where $r(s, a) > 0.01$. Calculated based on 100,000 samples from a random Gaussian policy with zero mean and unit standard deviation in each action dimension. Excludes 7 tasks where no learning occurs under any algorithm. Sorted from high to low.

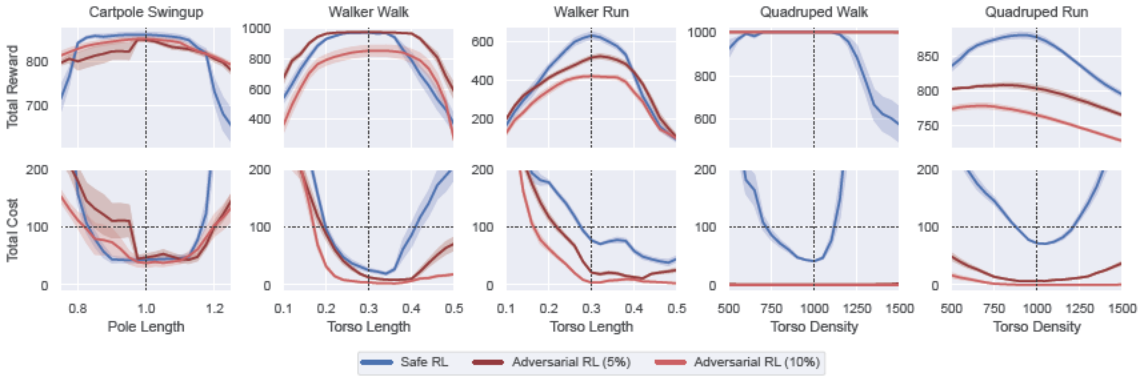

Figure C·5: Performance of adversarial RL across tasks and test environments. Performance of adversarial RL is evaluated without adversarial interventions. Shading denotes half of one standard error across policies. Vertical dotted lines represent nominal training environment. Top: Total reward. Bottom: Total cost, where horizontal dotted lines represent the safety budget and values below these lines represent safety constraint satisfaction.

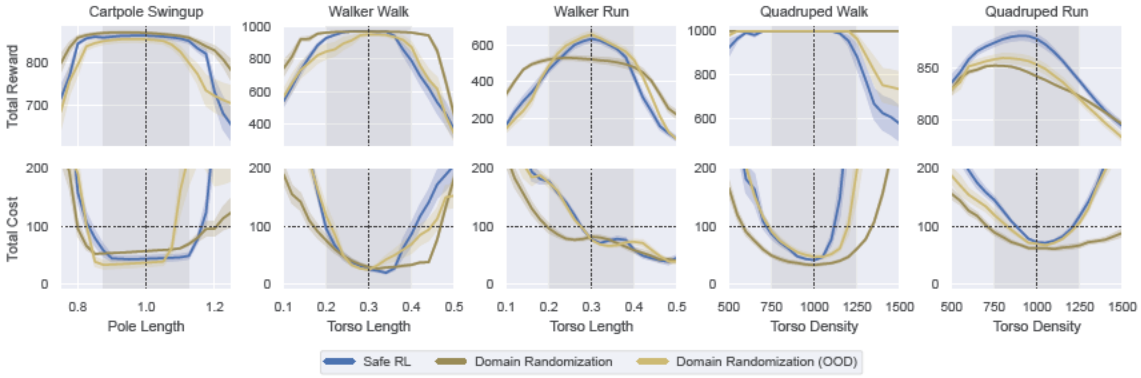

Figure C·6: Performance of domain randomization across tasks and test environments. Grey shaded areas represent the training distribution ranges for the in-distribution version of domain randomization. Shading denotes half of one standard error across policies. Vertical dotted lines represent nominal training environment. Top: Total reward. Bottom: Total cost, where horizontal dotted lines represent the safety budget and values below these lines represent safety constraint satisfaction.

References

Abdolmaleki, A., Huang, S., Hasenclever, L., Neunert, M., Song, F., Zambelli, M., Martins, M., Heess, N., Hadsell, R., and Riedmiller, M. (2020). A distributional view on multi-objective policy optimization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 11–22. PMLR.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. (2018). Maximum a posteriori policy optimisation. In *Sixth International Conference on Learning Representations*.

Abdullah, M. A., Ren, H., Ammar, H. B., Milenkovic, V., Luo, R., Zhang, M., and Wang, J. (2019). Wasserstein robust reinforcement learning. arXiv preprint. arXiv:1907.13196.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 22–31. PMLR.

Altman, E. (1999). *Constrained Markov Decision Processes*. CRC Press.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., and Bachem, O. (2021). What matters for on-policy deep actor-critic methods? A large-scale study. In *Ninth International Conference on Learning Representations*.

Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9(3):203–228.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 449–458. PMLR.

Blanchet, J. and Murthy, K. (2019). Quantifying distributional model risk via optimal transport. *Mathematics of Operations Research*, 44(2):565–600.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym. arXiv preprint. arXiv:1606.01540.

Cao, W., Benosman, M., and Ma, R. (2022). Domain knowledge-based automated analog circuit design with deep reinforcement learning. In *The 59th ACM/IEEE Design Automation Conference*.

Chen, R. and Paschalidis, I. C. (2020). Distributionally robust learning. *Foundations and Trends® in Optimization*, 4(1-2):1–243.

Chow, Y., Tamar, A., Mannor, S., and Pavone, M. (2015). Risk-sensitive and robust decision-making: a CVaR optimization approach. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1096–1105. PMLR.

de Bruin, T., Kober, J., Tuyls, K., and Babuška, R. (2018). Experience selection in deep reinforcement learning for control. *Journal of Machine Learning Research*, 19(9):1–56.

Deisenroth, M. and Rasmussen, C. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pages 465–472. ACM.

Derman, E., Mankowitz, D. J., Mann, T. A., and Mannor, S. (2018). Soft-robust actor-critic policy-gradient. arXiv preprint. arXiv:1803.04848.

Derman, E. and Mannor, S. (2020). Distributional robustness and regularization in reinforcement learning. arXiv preprint. arXiv:2003.02894.

Dhaene, J., Kukush, A., Linders, D., and Tang, Q. (2012). Remarks on quantiles and distortion risk measures. *European Actuarial Journal*, 2:319–328.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pages 1329–1338. PMLR.

Duisterhof, B. P., Krishnan, S., Cruz, J. J., Banbury, C. R., Fu, W., Faust, A., de Croon, G. C. H. E., and Janapa Reddi, V. (2021). Tiny robot learning (tinyRL) for source seeking on a nano quadcopter. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7242–7248.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. (2020). An empirical investigation of the challenges of real-world reinforcement learning. arXiv preprint. arXiv:2003.11881.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110:2419–2468.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. (2020). Implementation matters in deep RL: A case study on PPO and TRPO. In *Eighth International Conference on Learning Representations*.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. (2018). IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1407–1416. PMLR.

Fakoor, R., Chaudhari, P., and Smola, A. J. (2020). P3O: Policy-on policy-off policy optimization. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115, pages 1017–1027. PMLR.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H. S., Kohli, P., and Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1146–1155. PMLR.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1587–1596. PMLR.

Grossman, L. and Plancher, B. (2022). Just round: Quantized observation spaces enable memory efficient learning of dynamic locomotion. arXiv preprint. arXiv: 2210.08065.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. (2017a). Q-Prop: Sample-efficient policy gradient with an off-policy critic. In *5th International Conference on Learning Representations*.

Gu, S., Lillicrap, T., Turner, R. E., Ghahramani, Z., Schölkopf, B., and Levine, S. (2017b). Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1861–1870. PMLR.

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 3207–3214. AAAI Press.

Hoffman, M. W., Shahriari, B., Aslanides, J., Barth-Maron, G., Momchev, N., Sinopalnikov, D., Stańczyk, P., Ramos, S., Raichuk, A., Vincent, D., Hussenot, L., Dadashi, R., Dulac-Arnold, G., Orsini, M., Jacq, A., Ferret, J., Vieillard, N., Ghasemipour, S. K. S., Girgin, S., Pietquin, O., Behbahani, F., Norman, T., Abdolmaleki, A., Cassirer, A., Yang, F., Baumli, K., Henderson, S., Friesen, A., Haroun, R., Novikov, A., Colmenarejo, S. G., Cabi, S., Gulcehre, C., Paine, T. L., Srinivasan, S., Cowie, A., Wang, Z., Piot, B., and de Freitas, N. (2020). Acme: A research framework for distributed reinforcement learning. arXiv preprint. arXiv:2006.00979.

Hong, Z.-W., Shann, T.-Y., Su, S.-Y., Chang, Y.-H., Fu, T.-J., and Lee, C.-Y. (2018). Diversity-driven exploration strategy for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Hou, L., Pang, L., Hong, X., Lan, Y., Ma, Z., and Yin, D. (2020). Robust reinforcement learning with Wasserstein constraint. arXiv preprint. arXiv:2006.00945.

Hsu, D., Kakade, S., and Zhang, T. (2012). A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17.

Isele, D. and Cosgun, A. (2018). Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 3302–3309. AAAI Press.

Iyengar, G. N. (2005). Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280.

Jang, I., Kim, H., Lee, D., Son, Y.-S., and Kim, S. (2020). Knowledge transfer for on-device deep reinforcement learning in resource constrained edge computing systems. *IEEE Access*, 8:146588–146597.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jones, B. L. and Zitikis, R. (2003). Empirical estimation of risk measures and related quantities. *North American Actuarial Journal*, 7(4):44–54.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 267–274. Morgan Kaufmann Publishers Inc.

Keramati, R., Dann, C., Tamkin, A., and Brunskill, E. (2020). Being optimistic to be conservative: Quickly learning a CVaR policy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4436–4443.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823. Curran Associates, Inc.

Konda, V. and Tsitsiklis, J. (2000). Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Kong, A. (1992). A note on importance sampling using standardized weights. Technical Report 348, Department of Statistics, The University of Chicago. `https://victorelvira.github.io/papers/kong92.pdf`.

Kuang, Y., Lu, M., Wang, J., Zhou, Q., Li, B., and Li, H. (2022). Learning robust policy against disturbance in transition dynamics via state-conservative policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7247–7254.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. (2018). Model-ensemble trust-region policy optimization. In *Sixth International Conference on Learning Representations*.

L.A., P. and Fu, M. C. (2022). Risk-sensitive reinforcement learning via policy gradient search. *Foundations and Trends® in Machine Learning*, 15(5):537–693.

Laroche, R., Trichelair, P., and Combes, R. T. D. (2019). Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3652–3661. PMLR.

Li, L., Littman, M. L., Walsh, T. J., and Strehl, A. L. (2011). Knows what it knows: A framework for self-aware learning. *Machine Learning*, 82:399–443.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations*.

Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, S., Li, B., and Zhao, D. (2022). Constrained variational policy optimization for safe reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 13644–13668. PMLR.

Ma, X., Xia, L., Zhou, Z., Yang, J., and Zhao, Q. (2020). DSAC: Distributional soft actor critic for risk-sensitive reinforcement learning. arXiv preprint. arXiv:2004.14547.

Majumdar, A. and Pavone, M. (2020). How should a robot assess risk? Towards an axiomatic theory of risk in robotics. In *Robotics Research*, pages 75–84. Springer International Publishing.

Mankowitz, D. J., Calian, D. A., Jeong, R., Paduraru, C., Heess, N., Dathathri, S., Riedmiller, M., and Mann, T. (2021). Robust constrained reinforcement learning for continuous control with model misspecification. arXiv preprint. arXiv: 2010.10644.

Mankowitz, D. J., Levine, N., Jeong, R., Abdolmaleki, A., Springenberg, J. T., Shi, Y., Kay, J., Hester, T., Mann, T., and Riedmiller, M. (2020). Robust reinforcement learning for continuous control with model misspecification. In *Eighth International Conference on Learning Representations*.

Meng, W., Zheng, Q., Shi, Y., and Pan, G. (2022). An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):2223–2235.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529–533.

Mowlavi, S., Benosman, M., and Nabi, S. (2022). Reinforcement learning state estimation for high-dimensional nonlinear systems. In *Tenth International Conference on Learning Representations*.

Neuman, S. M., Plancher, B., Duisterhof, B. P., Krishnan, S., Banbury, C., Mazumder, M., Prakash, S., Jabbour, J., Faust, A., de Croon, G. C., and Reddi, V. J. (2022). Tiny robot learning: Challenges and directions for machine learning in resource-constrained robots. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 296–299.

Nilim, A. and Ghaoui, L. E. (2005). Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798.

Novati, G. and Koumoutsakos, P. (2019). Remember and forget for experience replay. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4851–4860. PMLR.

O'Donoghue, B., Munos, R., Kavukcuoglu, K., and Mnih, V. (2017). Combining policy gradient and Q-learning. In *5th International Conference on Learning Representations*.

Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. (2018). Stochastic variance-reduced policy gradient. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4026–4035. PMLR.

Paternain, S., Chamon, L., Calvo-Fullana, M., and Ribeiro, A. (2019). Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Pau, D., Colella, S., and Marchisio, C. (2023). End to end optimized tiny learning for repositionable walls in maze topologies. In *2023 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–7.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. (2017). Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2817–2826. PMLR.

Pydi, M. S. and Jog, V. (2020). Adversarial risk via optimal transport and optimal couplings. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 7814–7823. PMLR.

Queeney, J. and Benosman, M. (2023). Risk-averse model uncertainty for distributionally robust safe reinforcement learning. arXiv preprint. arXiv:2301.12593.

Queeney, J., Ozcan, E. C., Paschalidis, I. C., and Cassandras, C. G. (2023). Optimal transport perturbations for safe reinforcement learning with robustness guarantees. arXiv preprint. arXiv:2301.13375.

Queeney, J., Paschalidis, I. C., and Cassandras, C. G. (2021a). Generalized proximal policy optimization with sample reuse. In *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc.

Queeney, J., Paschalidis, I. C., and Cassandras, C. G. (2021b). Uncertainty-aware policy optimization: A robust, adaptive trust region approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9377–9385. AAAI Press.

Queeney, J., Paschalidis, I. C., and Cassandras, C. G. (2022). Generalized policy improvement algorithms with theoretically supported sample reuse. arXiv preprint. arXiv:2206.13714.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. (2017). EPOpt: Learning robust neural network policies using model ensembles. In *5th International Conference on Learning Representations*.

Rajeswaran, A., Mordatch, I., and Kumar, V. (2020). A game theoretic framework for model based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 7953–7963. PMLR.

Ray, A., Achiam, J., and Amodei, D. (2019). Benchmarking safe exploration in deep reinforcement learning. `https://cdn.openai.com/safexp-short.pdf`.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Russel, R. H., Benosman, M., Van Baar, J., and Corcodel, R. (2021). Lyapunov robust constrained-MDPs: Soft-constrained robustly stable policy optimization under model uncertainty. arXiv preprint. arXiv:2108.02701.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *4th International Conference on Learning Representations*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588:604–609.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1889–1897. PMLR.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint. arXiv:1707.06347.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2014). *Lectures on Stochastic Programming: Modeling and Theory, Second Edition*. Society for Industrial and Applied Mathematics.

Shen, Y., Tobia, M. J., Sommer, T., and Obermayer, K. (2014). Risk-sensitive reinforcement learning. *Neural Computation*, 26(7):1298–1328.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., Heess, N., Belov, D., Riedmiller, M., and Botvinick, M. M. (2020). V-MPO: On-policy maximum a posteriori policy optimization for discrete and continuous control. In *Eighth International Conference on Learning Representations*.

Stooke, A., Achiam, J., and Abbeel, P. (2020). Responsive safety in reinforcement learning by PID Lagrangian methods. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 9133–9143. PMLR.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Tamar, A., Chow, Y., Ghavamzadeh, M., and Mannor, S. (2015). Policy gradient for coherent risk measures. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Tessler, C., Efroni, Y., and Mannor, S. (2019a). Action robust reinforcement learning and applications in continuous control. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6215–6224. PMLR.

Tessler, C., Mankowitz, D. J., and Mannor, S. (2019b). Reward constrained policy optimization. In *Seventh International Conference on Learning Representations*.

Thomas, P., Theocharous, G., and Ghavamzadeh, M. (2015). High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2380–2388. PMLR.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. (2020). dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022.

Villani, C. (2008). *Optimal transport, old and new*. Springer.

Vinitsky, E., Du, Y., Parvate, K., Jang, K., Abbeel, P., and Bayen, A. (2020). Robust reinforcement learning using adversarial populations. arXiv preprint. arXiv:2008.01825.

Wang, C., Wu, Y., Vuong, Q., and Ross, K. (2020). Striving for simplicity and performance in off-policy DRL: Output normalization and non-uniform sampling. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 10070–10080. PMLR.

Wang, S. (1996). Premium calculation by transforming the layer premium density. *ASTIN Bulletin*, 26(1):71–92.

Wang, S. S. (2000). A class of distortion operators for pricing financial and insurance risks. *The Journal of Risk and Insurance*, 67(1):15–36.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2017). Sample efficient actor-critic with experience replay. In *5th International Conference on Learning Representations*.

Weber, P., Wälchli, D., Zeqiri, M., and Koumoutsakos, P. (2022). Remember and forget experience replay for multi-agent reinforcement learning. arXiv preprint. arXiv:2203.13319.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.

Wirch, J. L. and Hardy, M. R. (2003). Distortion risk measures: Coherence and stochastic dominance. *Insurance Mathematics and Economics*, 32:168–168.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xu, H. and Mannor, S. (2010). Distributionally robust Markov decision processes. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Xu, J., Kim, S., Chen, T., Garcia, A. R., Agrawal, P., Matusik, W., and Sueda, S. (2023). Efficient tactile simulation with differentiability for robotic manipulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205, pages 1488–1498. PMLR.

Xu, M., Liu, Z., Huang, P., Ding, W., Cen, Z., Li, B., and Zhao, D. (2022). Trustworthy reinforcement learning against intrinsic vulnerabilities: Robustness, safety, and generalizability. arXiv preprint. arXiv:2209.08025.

Xu, T., Liang, Y., and Lan, G. (2021). CRPO: A new approach for safe reinforcement learning with convergence guarantee. In *Proceedings of the 38th International Conference on Machine Learning*, pages 11480–11491. PMLR.

Yu, P. and Xu, H. (2016). Distributionally robust counterpart in Markov decision processes. *IEEE Transactions on Automatic Control*, 61(9):2538–2543.

CURRICULUM VITAE