Risk-Sensitive and Robust Model-Based Reinforcement Learning and Planning

Marc Rigter

Pembroke College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2022

Risk-Sensitive and Robust Model-Based Reinforcement Learning and Planning

Candidate: Marc Rigter
Supervisors: Professor Nick Hawes, Dr Bruno Lacerda
Examiners: Professor Jakob Foerster, Professor Aviv Tamar
Date of examination: 30th January, 2023

University of Oxford
Goal-Oriented Long-Lived Systems Group (GOALS)
Department of Engineering Science

Acknowledgements

I'm deeply grateful to my supervisors Nick and Bruno. Thank you for being fantastic mentors, and for always being generous with your time. This thesis would not have been possible without your guidance, knowledge, enthusiasm, and support.

I greatly appreciate the insightful feedback of my examiners, Jakob Foerster and Aviv Tamar, which has helped to significantly improve the final version of this thesis.

Many thanks to Benjamin Morrell, KC Wong, Rob Reid, and David Willson, for mentoring me when I first became interested in research. Thank you for having the patience and generosity to assist an enthusiastic undergraduate student.

I would like to extend my thanks to the members of the GOALS group that have been here throughout my DPhil: Charlie, Michael, Paul, Anna, Mohamed, Clarrisa, Matt, Raunak, Branton, Lara, and Alex. Thank you also to Luigi. I'm grateful to you all for your willingness to discuss ideas, and for making the environment we work in such a good one.

Thank you to Harrison Steel and Alex Kendall for taking the time to give me advice when I was applying to postgraduate studies when you had zero obligation to do so. I will try my best to pass the help forward to future students.

I'm grateful to Paul Billing and Lara Cordoni, among the other teachers at JPC, for first helping me to get excited about science and mathematics.

Thank you to my friends outside of work in New Zealand, Australia, and Oxford. In particular, I would like to thank Bobby, Renee, Taylor, Sam K, Robin, Andrew, Rhys, Sam C, Tim, Rob, Kacy, Owen, Gonzalo, Kayla, Faizal, Flaminia, David, Jose, and Juliana. I'm very lucky to know such awesome people.

Thank you Hannah for always being an amazing supporter, especially at the toughest points in this journey. I'm so glad that we have been able to go through our experience of Oxford together, and I'm very excited for the coming years.

I would like to express my gratitude to my grandmother, Lila, and to my sister, Katie. Grandma, thank you for always keeping me well fed, and for being one of the most joyful people I know. Katie, thank you for being a role model when we were younger, and for inspiring me to do more cool stuff outside of work.

Finally, I would like to thank my parents, Sandra and Tim. You have always been extremely supportive of everything I have tried to do, and I'm grateful for the lengths you went to in ensuring Katie and I had the best possible education. I would not have made it very far without your support.

Abstract

Many sequential decision-making problems that are currently automated, such as those in manufacturing or recommender systems, operate in an environment where there is either little uncertainty, or zero risk of catastrophe. As companies and researchers attempt to deploy autonomous systems in less constrained environments, it is increasingly important that we endow sequential decision-making algorithms with the ability to reason about uncertainty and risk.

In this thesis, we will address both planning and reinforcement learning (RL) approaches to sequential decision-making. In the planning setting, it is assumed that a model of the environment is provided, and a policy is optimised within that model. Reinforcement learning relies upon extensive random exploration, and therefore usually requires a simulator in which to perform training. In many real-world domains, it is impossible to construct a perfectly accurate model or simulator. Therefore, the performance of any policy is inevitably uncertain due to the incomplete knowledge about the environment. Furthermore, in stochastic domains, the outcome of any given run is also uncertain due to the inherent randomness of the environment. These two sources of uncertainty are usually classified as epistemic, and aleatoric uncertainty, respectively. The over-arching goal of this thesis is to contribute to developing algorithms that mitigate both sources of uncertainty in sequential decision-making problems.

We make a number of contributions towards this goal, with a focus on model-based algorithms. We begin by considering the simplest case where the Markov decision process (MDP) is fully known, and propose a method for optimising a risk-averse objective while optimising the expected value as a secondary objective. For the remainder of the thesis, we no longer assume that the MDP is fully specified. We consider several different representations of the uncertainty over the MDP, including a) an uncertainty set of candidate MDPs, b) a prior distribution over MDPs, and c) a fixed dataset of interactions with the MDP. In setting a), we propose a new approach to approximate the minimax regret objective, and find a single policy with low sub-optimality across all candidate MDPs. In b), we propose to optimise for risk-aversion in Bayes-adaptive MDPs to avert risks due to both epistemic and aleatoric uncertainty under a single framework. In c), the offline RL setting, we propose two algorithms to overcome the uncertainty that stems from only having access to a fixed dataset. The first proposes a scalable algorithm to solve a robust MDP formulation of offline RL, and the second approach is based on risk-sensitive optimisation. In the final contribution chapter, we consider an interactive formulation of learning from demonstration. In this problem, it is necessary to reason about uncertainty over the performance of the current policy, to selectively choose when to request demonstrations. Empirically, we demonstrate that the algorithms we propose can generate risk-sensitive or robust behaviour in a number of different domains.

Contents

1 Introduction

In the coming years we will inevitably rely more on algorithms for decision-making, in applications ranging from robotics and self-driving cars [1], to healthcare [2], smart grids [3], and finance [4]. Indeed, one study estimates that 47% of current jobs are likely to be automated in the next two decades [5].

Many processes that are currently automated, such as those in manufacturing or online advertising, operate in an environment where there is either little uncertainty, or zero risk of catastrophe. As companies and researchers attempt to deploy autonomous systems in less constrained environments, it is increasingly important that we endow algorithms with the ability to reason about uncertainty and risk in a manner that reflects our priorities.

As a motivating example, consider a healthcare setting where the goal is to recommend a treatment plan for a patient. Due to variations between patients, it is impossible to predict the exact outcome of a treatment plan for any given patient. An aggressive treatment plan may have the best outcome for most patients, yet result in adverse consequences for a minority. We may prefer to avoid these adverse consequences altogether by using a less aggressive treatment plan with lower risk, even if this results in slightly worse outcomes for the majority of patients. An even better alternative could be to give each patient the opportunity to decide which treatment plan they would prefer, given their own outlook on the risks involved.

Another application in which it is necessary to reason about uncertainty and risk is robotics. Robotic systems that interact with the human world, such as self-driving cars, must be robust to almost all situations that they might encounter, not just the most common ones. As argued by [6], the knowledge that a self-driving

car typically avoids crashes is unlikely to be reassuring to passengers or regulators. Instead, it would be more convincing to know that the car had been designed to act conservatively to avoid catastrophe even in unlikely situations, at the expense of making the system slower or less versatile.

The goal of this thesis is to contribute towards the development of algorithms that can reason about uncertainty and risk in a flexible manner. We focus on *sequential* decision-making problems, where the decision made at one point in time influences the outcomes and possible courses of action in the future. Markov decision processes (MDPs) are a standard framework for sequential decision-making under uncertainty, and are the model that we will consider throughout this thesis. The vast majority of research for MDPs develops algorithms that optimise performance in *expectation*. For most applications, reasoning about the expectation is sufficient. However, as we have discussed, in safety-critical settings we must also consider the *variability* in the outcomes.

In this thesis, we will predominantly focus on model-based approaches that model the dynamics of the environment, and use this model to inform the sequence of decisions that should be made. In contrast to model-free approaches, model-based approaches are more suitable for guaranteeing safety [7], and can be used to improve interpretability [8]. Furthermore, recent work has shown that it is possible for model-based methods to match the performance of model-free methods, even for complex tasks [9].

With the goal of designing algorithms that reason about uncertainty in mind, there are two key variables that we must consider: a) which source or sources of uncertainty would we like to address, and b) what type of objective we would like to optimise in the face of this uncertainty. The source of uncertainty can be classified as *epistemic* or *aleatoric* uncertainty, and the objectives that we may wish to optimise for can be classified as *risk-sensitive* or *robust* approaches. These concepts guide the development of most of the work in this thesis. Furthermore, depending on the context, we may wish to develop an approach based on *planning* or *reinforcement learning* algorithms. This thesis considers both paradigms. In

the remainder of this section, we will provide an overview of each of these key concepts and how they relate to the research in this thesis.

Epistemic and aleatoric uncertainty Sources of uncertainty are usually categorised into two types: epistemic uncertainty, which stems from a lack of knowledge about a parameter or phenomenon; and aleatoric uncertainty, which stems from the inherent variability in a random event. Epistemic uncertainty can be reduced by gathering more data about the unknown phenomenon, whereas aleatoric uncertainty cannot. In the context of control problems such as MDPs, epistemic and aleatoric uncertainty are also referred to as parametric and internal uncertainty respectively [10].

In model-based algorithms for MDPs, epistemic uncertainty manifests in uncertainty over the transition and/or reward function of the MDP. As more data is collected from interacting with the MDP, our knowledge about the MDP dynamics improves and the epistemic uncertainty is reduced. Aleatoric uncertainty manifests in the stochastic transition and/or reward function of the MDP. Even if we have perfect knowledge of the MDP, there is still variability over the possible outcomes for each episode due to this inherent stochasticity. Throughout this thesis, we explore approaches that mitigate either epistemic or aleatoric uncertainty, or a combination of both sources of uncertainty.

Risk-sensitivity and robustness Approaches to reasoning about and mitigating uncertainty can generally be placed into one of two different categories. *Robust* approaches take a Knightian view of uncertainty [11], and do not require the relative likelihood of possible outcomes to be quantified. In this type of approach, it is assumed that there is a set of plausible scenarios (the so-called "uncertainty" or "ambiguity" set). The goal is usually to achieve the best possible performance in the worst-case instantiation of the uncertainty. In other words, this approach ensures that the solution is "robust" to all possible scenarios.

In the context of MDPs, the uncertainty set is usually defined as a set of plausible MDPs. The most common objective is to find a policy that has the best expected value for the worst possible MDP within the set [12, 13]. This approach is averse

to the epistemic uncertainty over the MDP. However, due to the consideration of expected value in each MDP, it is neutral towards aleatoric uncertainty. This is a natural formulation for problems where the policy is executed in the MDP many times, so that the aleatoric risk (i.e. the fluctuation in performance between each run) averages out [14].

A shortcoming of the robust optimisation perspective is that it can be overly conservative: performance may be sacrificed in the vast majority of cases for the sake of improving performance in a highly unlikely worst-case situation. Furthermore, the robust approach only takes into consideration the epistemic uncertainty and therefore does not consider the risk of a poor outcome for any given run.

The alternative is to consider a *risk-sensitive* approach. Here, the goal is to optimise a risk measure that maps the distribution over outcomes to a scalar value. Thus, unlike robust optimisation approaches, risk-sensitive approaches reason about the likelihood of each outcome. Depending on the risk measure, the variability of the outcome, and the likelihood of catastrophic outcomes, are penalised to a varying degree. Depending on the appetite for risk, an optimal risk-sensitive strategy may be willing to accept a small probability of an adverse outcome if it improves performance in most scenarios.

In this thesis, we consider robust approaches that optimise for the worst-case scenario to mitigate epistemic uncertainty in Chapters 4 and 6. We also propose risk-sensitive approaches that optimise a risk measure to address aleatoric uncertainty (Chapter 3) as well as epistemic uncertainty (Chapters 5 and 7). Our work on risk-sensitive algorithms will focus on generating *risk-averse* behaviour, i.e. avoiding risks. This is in contrast to *risk-seeking* behaviour, which may be useful for exploration [15].

Planning and reinforcement learning We consider algorithmic approaches related to both planning and reinforcement learning. In the planning setting, it is assumed that a model of the environment is known and we are interested in finding a suitable policy given that model. In the reinforcement learning setting,

a model of the environment is not provided a priori. Instead, we derive a policy from samples of interaction with the environment.

Reinforcement learning methods can either be model-free, meaning that they do not require a model of the environment, or model-based, meaning that a model is learnt from environment interaction data, and this learnt model is utilised when optimising the policy. The work on reinforcement learning in this thesis is predominantly model-based.

Reinforcement learning can further be categorised into online and offline variants. In online reinforcement learning, further exploration can be used to gather more samples from the environment. In offline, or batch reinforcement learning, the dataset of interactions with the environment is fixed.

During the first half of this thesis, we focus on *tabular* planning methods. These methods are suitable for problems where there is a small discrete state and action space. In the latter half of this thesis, we leverage deep reinforcement learning techniques that use neural network function approximation to enable greater scalability.

1.1 Contributions

This thesis contributes several approaches to solving sequential decision-making problems that go beyond optimising for the expected performance. Each chapter considers a different combination of the concepts outlined in the previous section: whether to address aleatoric uncertainty, epistemic uncertainty, or both; whether to consider a risk-sensitive objective or a robust one; and whether the model is provided (planning) or learnt from interactions with the environment (model-based reinforcement learning). In Chapters 3, 4, 6, and 7 we develop more performant algorithms for existing problem formulations. In Chapters 5 and 8 we propose new problem formulations in addition to algorithms for those problems. Chapters 3-5 address tabular MDPs, while Chapters 6-8 present more scalable approaches that utilise function approximation.

Here, we provide a summary of the contributions of each of the papers that comprise this thesis in the order that they are presented in Chapters 3-8. We begin by considering the simplest case where a tabular MDP is assumed to be known exactly. We focus on the tradeoff between optimising for a risk-sensitive objective and the expected performance. Specifically, we consider conditional value at risk (CVaR) which will be elaborated upon in the following chapter. We make the following contributions:

- Showing that there can be multiple policies that obtain the optimal CVaR in an MDP.
- An algorithm for optimising expected value in MDPs subject to the constraint that CVaR is optimal.

These contributions are presented in Chapter 3: *Planning for Risk-Aversion and Expected Value in MDPs*. This work makes the strong assumption that the MDP is known exactly, i.e. that there is no epistemic uncertainty. Throughout the remainder of the thesis, we no longer make this restrictive assumption.

The *uncertain MDP* setting considers uncertainty over the MDP represented in the form of an uncertainty set of plausible MDPs, any one of which could be the true MDP. The *minimax regret* objective aims to find a single policy that has the lowest sub-optimality, in terms of expected value, over the set of possible MDPs.

In our work on the uncertain MDP setting, we make the following contributions:

- A new approach to approximating the minimax regret objective.
- Proposing the use of options to capture dependencies between uncertainties to trade solution quality against the computation required.

These contributions are presented in Chapter 4: *Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes.* In an earlier version of this work, we incorrectly claimed that our approach is exact under certain assumptions. This error is discussed in Chapter 4, and corrected in the associated corrigendum.

In the uncertain MDP setting in Chapter 4, we assumed that we cannot learn more information about the environment through interaction. In the next chapter, we consider the Bayesian formulation of model-based RL in which we maintain a belief distribution over the unknown MDP. The belief is refined as experience is collected within the environment, thus enabling the policy to adapt to the true underlying environment. This problem can be formulated as planning in a Bayes-adaptive MDP (BAMDP), a special class of Partially Observable MDP (POMDP), in which the state is augmented with the belief over the underlying MDP.

In this problem setting, we make the following contributions:

- Proposing to optimise the CVaR of the return in the BAMDP as a means to mitigate both epistemic and aleatoric uncertainty under a single framework.
- An algorithm based on MCTS and Bayesian optimisation for this problem.

These contributions are presented in Chapter 5: *Risk-Averse Bayes-Adaptive Reinforcement Learning.*

Chapter 4 and Chapter 5 assume that prior information about the MDP is specified either in the form of a set of plausible MDPs, or a suitable prior over MDPs, respectively. In Chapters 6 and 7 we address the offline deep reinforcement learning setting, in which the starting point is a dataset of transitions collected from the environment. Unlike the standard RL setting, no further exploration of the environment is allowed. We consider model-based offline RL, where we first learn a model of the environment dynamics from the dataset, and optimise a policy within that model. The naive application of this approach results in the policy learning to exploit areas of the model not covered by the dataset, where the model is inaccurate. This problem is referred to as distributional shift. In Chapter 6, *RAMBO: Robust Adversarial Model-Based Offline RL*, we propose a novel approach to address this issue, and make the following contributions:

- Introducing a novel and theoretically-grounded model-based offline RL algorithm which enforces conservatism and prevents distributional shift by

training an adversarial dynamics model to generate pessimistic synthetic data for out-of-distribution actions.

- Adapting Robust Adversarial RL (RARL) to model-based offline RL by proposing a new formulation of RARL, where instead of defining and training an adversary policy, we directly train the model adversarially.

In Chapter 7, *One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline RL*, we propose a risk-sensitive approach to offline RL. In this chapter, we utilise risk-sensitive optimisation to avoid the issue of distributional shift. This work makes the following contributions:

- Proposing risk-sensitive optimisation as a method to address the issue of distributional shift in offline RL.
- Introducing the first model-based algorithm for risk-sensitive offline RL, which jointly addresses epistemic and aleatoric uncertainty.

The main motivation for the offline RL setting addressed in Chapters 6 and 7 is avoiding the extensive online exploration required by online RL. In the last contribution chapter, we take an alternative approach to reducing the amount exploration required, and consider learning from demonstration. In our interactive setting, demonstrations can be requested from an expert to help train a policy that is further fine-tuned using RL.

In this work, we make the following contributions:

- A new problem formulation where there is a cost associated with the human effort for providing demonstrations, and resetting the system from failure. The goal is to minimise the total human cost incurred over many episodes.
- An algorithm for deciding when to request demonstrations which is based on posing the problem as a contextual multi-armed bandit.

This is presented in Chapter 8: *A Framework for Learning from Demonstration with Minimal Human Effort.*

1.1.1 Contribution Summary

Table 1.1 summarises the attributes of each chapter in this thesis. Each chapter is summarised according to the following attributes:

- **Prior Knowledge** At one end of the spectrum, Chapter 3 assumes that we have perfect knowledge of the environment, while at the other end, Chapters 6 and 7 assume that we only have access to transition data from the environment which may be noisy or of low quality. The other chapters make different assumptions: that the uncertainty over the MDP is represented by an uncertainty set or prior distribution, or that we have access to demonstrations from an expert.

- **Considering Aleatoric Uncertainty** Approaches that consider aleatoric uncertainty take into account the variability in the performance of the policy due to inherent stochasticity in the environment. Approaches which optimise for the expected value disregard aleatoric uncertainty. Chapters 3, 5, and 7 take aleatoric uncertainty into consideration by optimising a risk measure.

- **Considering Epistemic Uncertainty** Approaches that consider epistemic uncertainty take into consideration the presence of incomplete knowledge about the environment due to modelling errors, or a lack of data. With the exception of Chapter 3, all the chapters of this thesis consider epistemic uncertainty in some manner.

- **Risk or Robustness** Risk-sensitive approaches optimise a risk measure, while robust approaches optimise for the worst-case scenario over an uncertainty set. With the exception of Chapter 8, all the chapters of this thesis can be classified into one of these two categories.

- **Paradigm** This attribute describes whether the work addresses tabular MDPs, or proposes an approach that utilises function approximation to improve scalability to large and/or continuous problems. The function approximators that we consider in this work are predominantly neural networks. The first

Table 1.1: Summary of the attributes of each chapter.

	Chapter 3	**Chapter 4**	**Chapter 5**	**Chapter 6**	**Chapter 7**	**Chapter 8**
Prior knowledge	MDP fully known	MDP uncertainty set	Prior over MDPs	Fixed dataset	Fixed dataset	Expert demonstrator
Considers aleatoric uncertainty	Yes	No	Yes	No	Yes	No
Considers epistemic uncertainty	No	Yes	Yes	Yes	Yes	Yes
Risk or robustness	Risk	Robustness	Risk	Robustness	Risk	N/A
Paradigm	Tabular	Tabular	Tabular	Function Approx.	Function Approx.	Function Approx.

Chapter 3 - Planning for Risk-Aversion and Expected Value in MDPs.
Chapter 4 - Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes.
Chapter 5 - Risk-Averse Bayes-Adaptive Reinforcement Learning.
Chapter 6 - RAMBO-RL: Robust Adversarial Model-Based Offline Reinforcement Learning.
Chapter 7 - One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline RL.
Chapter 8 - A Framework for Learning from Demonstration with Minimal Human Effort.

half of this thesis considers tabular MDPs, while the latter half proposes approaches that utilise function approximation.

In each chapter of the thesis, we repeat Table 1.1 to discuss how the ideas proposed in that chapter relate back to each of the attributes described here.

1.2 Thesis Structure

The following chapter is a literature review providing context for this thesis. As this is an integrated thesis, each of the subsequent chapters corresponds to a paper. In each of the paper chapters, we first give an overview of the contributions of the paper. Following the presentation of each paper, we discuss the limitations of the work and suggest directions for future work to address these limitations. The full appendices of each paper are included at the end of this document. However, in the contribution chapters we have included portions of the appendices that provide additional insight into our work. Citations in the main body of the thesis refer to references included

in the bibliography at the end of this thesis, following Chapter 9. Citations within each paper refer to the references in the bibliography at the end of that paper.

In chronological order, the publications which comprise this integrated thesis are the following [16, 17, 18, 19, 20, 21]:

- Marc Rigter, Bruno Lacerda, and Nick Hawes (2020). A framework for learning from demonstration with minimal human effort. *IEEE Robotics and Automation Letters* (RAL).
- Marc Rigter, Bruno Lacerda, and Nick Hawes (2021). Minimax regret optimisation for robust planning in uncertain Markov decision processes. *AAAI Conference on Artificial Intelligence* (AAAI).
- Marc Rigter, Bruno Lacerda and Nick Hawes (2021). Risk-averse Bayes-adaptive reinforcement learning. *Advances in Neural Information Processing Systems* (NeurIPS).
- Marc Rigter, Paul Duckworth, Bruno Lacerda and Nick Hawes (2022). Planning for risk-aversion and expected value in MDPs. *International Conference on Automated Planning and Scheduling* (ICAPS).
- Marc Rigter, Bruno Lacerda and Nick Hawes (2022). RAMBO-RL: Robust adversarial model-based offline reinforcement learning. *Advances in Neural Information Processing Systems* (NeurIPS).
- Marc Rigter, Bruno Lacerda and Nick Hawes (2022). One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline Reinforcement Learning. *arXiv preprint arXiv:2212.00124*.

Note that to improve the flow of the thesis, the papers are not presented in chronological order. Other publications that I have contributed to over the course of my DPhil that are not included in this thesis are [22, 23, 24, 25, 26]:

- Shu Ishida, Marc Rigter, and Nick Hawes. Robot path planning for multiple target regions (2019). *European Conference on Mobile Robots* (ECMR).

- Marc Rigter, Danial Dervovic, Parisa Hassanzadeh, Jason Long, Parisa Zehtabi, Daniele Magazzeni (2022). Optimal admission control for multiclass queues with time-varying arrival rates. *AAAI Conference on Artificial Intelligence* (AAAI).

- Clarissa Costen, Marc Rigter, Bruno Lacerda and Nick Hawes (2022). Shared autonomy systems with stochastic operator models. *International Joint Conference on Artificial Intelligence* (IJCAI).

- Anna Gautier, Marc Rigter, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. (2022). Risk-Constrained Planning for Multi-Agent Systems with Shared Resources. *International Conference on Autonomous Agents and Multiagent Systems*.

- Clarissa Costen, Marc Rigter, Bruno Lacerda and Nick Hawes (2023). Planning with hidden parameter polynomial MDPs. *AAAI Conference on Artificial Intelligence* (AAAI).

2 Background

2.1 Markov Decision Processes

Markov decision processes (MDPs) [27, 28] are a standard framework for sequential decision-making under uncertainty. An MDP is defined by the tuple $M = (S, A, T, R, \mu_0, \gamma)$. S defines the set of possible states of the environment, A defines the set of actions available to the agent, and μ_0 defines the initial state distribution. At each step, the agent observes the current state, $s \in S$, and on that basis chooses an action, $a \in A$. After applying the action, the agent receives a reward according to the reward function, $R : S \times A \to \mathbb{R}$ (or equivalently a cost). Subsequently, the environment transitions stochastically to a new state according to the transition function, $T : S \times A \to \Delta S$, where ΔS indicates a distribution over states.

A deterministic Markovian policy is a mapping from each state to an action, $\pi : S \to A$. The standard objective in MDPs is to find the policy, π^*, that maximises the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}^{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big]. \tag{2.1}$$

2.1.1 Solution Methods

Value-based methods for MDPs are based on the principle of dynamic programming [29], which decomposes the overarching optimisation problem into a sequence of sub-problems. These methods solve Equation 2.1 by computing the optimal *value function* at each state:

$$V^*(s) = \max_{\pi} \Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s \Big],$$

which can be found by recursively applying the Bellman equation:

$$V^*(s) = \max_{a} \Big[R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) \cdot V^*(s') \Big].$$

Exact planning approaches [30, 27] compute the value function exactly. In cases where the transition dynamics are unknown or state space is large, approximate methods such as sample-based planning [31] or reinforcement learning (RL) [28] are more suitable. These latter approaches usually approximate the value function by sampling interactions with the environment, although some RL methods forgo learning a value function in favour of directly optimising the policy [32].

2.1.2 Overview of Related Work

The majority of algorithms for MDPs optimise the expected value objective given in Equation 2.1, and ignore the variability in the cost or reward received between each episode. As discussed in Chapter 1, for some applications, ignoring variability in the outcomes is inappropriate. Some degree of sensitivity towards adverse outcomes can be introduced into the expected reward framework by shaping the reward function by adding large negative rewards. However, "hacking" the reward function in this manner magnifies the already difficult problem of reward function design [33]. If the reward function is not designed carefully, this can lead to irrational behaviour [34]. Furthermore, shaping the reward function can make it difficult to understand what the algorithm is ultimately optimising for [35], especially in domains where the original reward function has a clear interpretation. Therefore, in Section 2.2 we will review risk-sensitive approaches that take into account variability in performance via the optimisation of risk measures which map the distribution over rewards to a scalar value.

An additional limitation of planning-based approaches is that they assume access to an exact model of the MDP. However, in practical applications the MDP cannot be estimated exactly, and even small errors in the model can result in

significant errors in the solution [10]. In Section 2.3, we review approaches that mitigate errors in the MDP model via worst-case approaches inspired by robust optimisation [12, 13]. Furthermore, in Section 2.4 we will consider approaches which produce a policy that adapts to the true underlying MDP.

Analogous to the requirement for a model for planning, RL methods usually require an accurate simulator [36]. This is because RL algorithms require extensive exploration to find a solution, which can be costly or dangerous in the real environment [37]. Therefore, performing online RL in the real environment is infeasible in many domains. An alternative is to instead consider the *offline* RL setting where a policy is trained using a fixed dataset with no online exploration. By optimising the reward signal, offline RL algorithms can improve upon the behaviour that generated the dataset. Offline RL is applicable to domains such as robotics [38] and healthcare [39], where it is possible to leverage large existing historical datasets. The main limitation of offline RL is that if there is no high-quality data in the dataset, it is not possible to learn a performant policy due to the absence of further exploration. We provide a more detailed review of offline RL in Section 2.5.

Another means to overcome the extensive exploration required by online RL is to use methods based on learning from demonstration [40] (also known as imitation learning). Unlike the offline RL setting, learning from demonstration assumes that the interaction data is collected using an expert policy. The policy derived from the demonstrations can be used as the final policy, or alternatively, if a reward function is available it can be further fine-tuned using RL [41]. We provide a brief overview of learning from demonstration in Section 2.6.

2.2 Risk in MDPs

In this subsection, we review previous work on risk in MDPs. We give an overview of risk measures, before discussing algorithms that optimise for risk in MDPs based on both planning or reinforcement learning. Throughout this subsection we will discuss risk in the context of mitigating aleatoric uncertainty due to the

inherent stochasticity of the environment. We will address epistemic uncertainty in future subsections.

For this subsection, we will assume that the objective is to minimise cost, as this is the standard convention when analysing risk. However, reward maximisation can equivalently be considered.

Risk Measures Early work on risk in MDPs [42] considers the exponential utility framework [43], where the cost is transformed according to a convex utility function to achieve risk-aversion. The expected value of this transformed cost is optimised. This generates risk-averse behaviour by magnifying the influence of large costs. However, as previously noted in this chapter, it is difficult to "shape" the cost function appropriately to achieve the desired behaviour [6].

A *risk measure* maps a probability distribution over possible costs to a scalar value [44]. Such measures take into consideration the variability of the cost. Depending on the risk measure, a varying degree of emphasis can be placed on the worst possible outcomes.

Denote by Ω the set of all possible outcomes in a probability space. Consider a function $Z : \Omega \to \mathbb{R}$ that assigns a cost of $Z(\omega)$ to any stochastic outcome $\omega \in \Omega$. Z is then a random variable representing the cost incurred depending on the outcome. A risk measure, ρ, is a mapping from any cost random variable, Z, to a real number.

Coherent risk measures are proposed by [45], and [6] argues that coherent risk measures should be employed when designing autonomous systems. In short, coherent risk measures adhere to several axioms associated with rational decision-making. The axioms that any coherent risk measure must adhere to are the following [45]:

- *Monotonicity.* Suppose that Z_1 and Z_2 are two cost random variables, and that $Z_1(\omega) \leq Z_2(\omega)$ for all $\omega \in \Omega$. Then $\rho(Z_1) \leq \rho(Z_2)$.

  Interpretation: if under all possible scenarios the cost for Z_1 is less than Z_2, then Z_1 should be considered less risky.

- *Translation invariance.* Consider some cost random variable, Z, and fixed cost $c \in \mathbb{R}$. Then $\rho(Z - c) = \rho(Z) - c$.

  Interpretation: a guaranteed reduction in the cost by a fixed amount reduces the risk by the same amount.

- *Positive homogeneity.* Let Z be some cost random variable, and $\beta \geq 0$ a scalar. Then $\rho(\beta Z) = \beta\rho(Z)$.

  Interpretation: if all costs are scaled by some amount, then the risk is scaled by that same amount.

- *Subadditivity.* Consider two cost random variables, Z_1 and Z_2. Then $\rho(Z_1 + Z_2) \leq \rho(Z_1) + \rho(Z_2)$.

  Interpretation: diversification by jointly taking two different strategies should be considered at most as risky as the individual strategies considered separately.

Some works on risk in MDPs consider the mean-variance criterion [46, 47] where the mean of the total cost is penalised by the variance. However, this is not a coherent risk measure as it is translation invariant but does not satisfy the other required axioms [6]. As an illustration of why this may lead to irrational behaviour, consider a reduction in the cost associated with all possible outcomes. Because the mean-variance criterion does not adhere to the monotonicity property, this is not guaranteed to improve the mean-variance objective. This is because even if the cost is reduced for all outcomes, the variance may increase, and outweigh the improvement in the mean.

Another risk measure that is not coherent is the *value at risk* (VaR). VaR is frequently used in financial decision-making [48]. For some probability level, the VaR is equal to the corresponding quantile of the cost distribution. VaR is not coherent as it does not satisfy the subadditivity property. In other words, the VaR of jointly taking two strategies can be higher than the sum of the VaR of the individual strategies. This problem is caused by the fact that the VaR is a

quantile and does not take into account the distribution over costs in the tail [49]. Because the distribution of the tail outcomes is not taken into consideration, VaR may under-represent the risk of the worst outcomes for fat-tailed distributions [50].

Risk measures which are coherent include the conditional value at risk (CVaR) [51], the entropic value at risk [52], the mean-semideviation [53], and the Wang risk measure [54]. Throughout this thesis, we focus predominantly on CVaR as it is intuitive to understand and is the risk measure most commonly used in recent research on both planning under uncertainty and reinforcement learning [55, 56, 57, 58]. It is also the risk measure recommended by central banks for assessing financial risk [59]. For a continuous cost random variable, the CVaR at confidence level α is the mean of the α-portion of the right tail of the cost distribution [57], as illustrated in Figure 2.1.

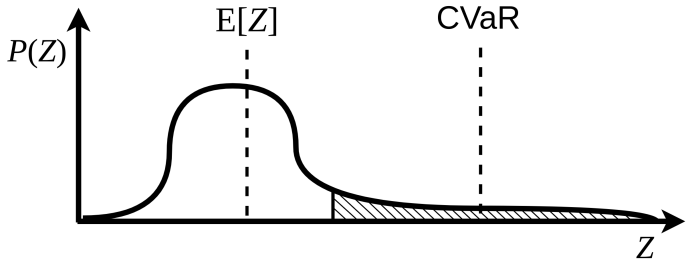


Figure 2.1: Illustration of the conditional value at risk at confidence level α for cost random variable Z. The shaded region indicates the α-portion of the tail of the distribution.

Static and Dynamic Risk There are two natural approaches to considering risk in sequential problems. The first is the *static* risk perspective, where we treat the total *cumulative* cost to be the random variable of interest [56]. Thus, risk is evaluated by applying a risk measure to the distribution over the cumulative cost. The main drawback of the static perspective for sequential problems is that the evaluation of static risk is not *time-consistent* [60]. This means that if the risk of an action is evaluated to be riskier at one point time, then it is not necessarily evaluated to be more risky at all other points in time. This leads to the optimal policy for static risk objectives being non-Markovian [58, 56].

The alternative approach is to consider the *dynamic* perspective of risk. In this approach, the risk measure is applied recursively at each point in time [61].

Thus, risk is evaluated by applying a risk measure to the distribution over outcomes at *every* time step. A formal definition of static and dynamic risk can be found in the preliminaries section of Chapter 7.

To illustrate the difference between static and dynamic risk, consider the conditional value at risk for the example illustrated in Figure 2.2. The static conditional value at risk is the expected value of the worst α-portion of runs. In the example, reaching any of the four final states is equally likely. We can compute static conditional value at risk for $\alpha = 0.5$ by computing the average cost on the worst 50% of runs. The worst 50% of runs are those which obtain a cost of either 2 or 4, and therefore the static conditional value at risk in the example is 3. In contrast, the dynamic conditional value at risk for $\alpha = 0.5$ is the expected value assuming that the worst 50% of transitions occur at each time step. This is the expected value assuming that transition A occurs at both time steps, and therefore the dynamic conditional value at risk for $\alpha = 0.5$ is 4.

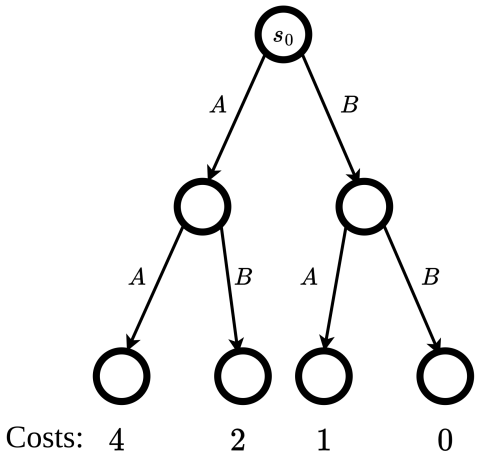


Figure 2.2: Illustration of a Markov chain (i.e. an MDP with no actions). At each state, transition A or transition B occurs with equal probability of 0.5. At the final state, the annotated cost is received.

The advantage of the dynamic risk perspective is that it is time-consistent and therefore the optimal policy is always Markovian [61]. A drawback of dynamic risk is that because it is evaluated recursively, it compounds the risk of poor outcomes at every step. Therefore, it can be much more conservative than the static risk approach and it is also more difficult to interpret [62].

Algorithms Algorithms for optimising dynamic risk include exact dynamic programming approaches [61], as well as approaches based on policy gradients [53]. For the static CVaR setting, algorithms based on value iteration over an augmented state space have been proposed [58, 56]. Other works use policy gradient methods to find a locally optimal solution for CVaR [63, 55, 57, 53, 64, 65, 66], or general coherent risk metrics [67]. A provably sample-efficient RL algorithm for optimising CVaR based on optimism in the face of uncertainty is presented in [68].

An alternative approach is taken by methods based on *distributional* reinforcement learning [69, 70]. Rather than estimating the expected value, these methods instead learn the distribution over the return. With access to a prediction of the full return distribution associated with each action, the policy can then be optimised for a static risk objective [71, 72, 73].

Relevance to this thesis In this thesis, we propose and address new formulations of risk-sensitive optimisation in both fully-known MDPs, and MDPs which are partially known. The fully specified MDP case is considered in Chapter 3, and the partially known MDP case is addressed in Chapters 5 and 7.

2.3 Robustness in MDPs

The previous section discussed risk-sensitivity towards the inherent stochasticity of the MDP. This addresses scenarios where either the MDP is known (for planning), or unlimited experience can be gathered in the true MDP (for reinforcement learning). However, in many cases we do not have access to the true MDP and have to optimise a policy using an imperfect approximation of the MDP. Therefore, we may wish to mitigate epistemic (also called parametric) uncertainty due to inexact information about the MDP.

One approach to doing this is to find a robust solution that performs well in all possible instantiations of the uncertainty [74]. *Uncertain* MDPs [75] assume that the true transition function for the MDP is known to lie within an uncertainty set. The *robust* solution to uncertain MDPs [13, 12, 75], also known as the Robust

MDP approach, optimises for the best expected cost in the worst-case instantiation of the MDP within the uncertainty set. This can be optimised efficiently with robust dynamic programming provided that: a) the uncertainty set is convex, and b) the uncertainty set is *rectangular*, meaning that the uncertainties are independent between each state-action pair [12, 13, 76] (see the preliminaries section of Chapter 4 for a more formal definition of rectangular uncertainties). The first assumption enables the worst-case transition function to be found efficiently at each state-action pair, and the latter assumption means that the problem admits a dynamic programming decomposition. Note that because the expected value is considered within each plausible MDP, this approach does not address aleatoric uncertainty.

This exact dynamic programming approach for Robust MDPs is applicable to problems with small state spaces. To scale up Robust MDPs to larger problems, algorithms which utilise reinforcement learning with function approximation have also been proposed [77, 78].

Optimising for the worst-case expected value often results in overly conservative policies [79]. This problem is exacerbated by the rectangularity assumption which allows the transition function for all state-action pairs to be realised as their worst-case values simultaneously. To avoid the assumption that all transition probabilities can take on their worst-case values, several alternative modelling approaches have been proposed. The "lightning does not strike twice" model [80, 14] assumes that the MDP parameter values deviate from nominal values a maximum number of times. An alternative approach is to represent the uncertainty over the MDP using a discrete set of plausible MDPs [81, 82, 83, 84, 85, 86]. We refer to this approach as *sample-based* uncertain MDPs. This enables the uncertainty set to represent correlations between the uncertainties at different state-action pairs. However, the sample-based approach drastically increases the computational complexity of the problem as it prohibits dynamic programming [83, 86].

Another option is to consider the *minimax regret* optimisation objective as an alternative to the worst-case objective discussed thus far [87]. In this context, minimax regret refers to minimising the maximum expected sub-optimality of a

single policy across any of the possible instantiations of the true MDP. This is less conservative than the worst-case robust objective, as to achieve low maximum regret the policy must be near-optimal across all possible instantiations of the MDP. In contrast, for the standard worst-case robust objective, the policy must only perform well in the worst-case MDP.

Regret is used to measure sample-efficiency in the context of RL theory (e.g. [88, 89, 90]). In the RL setting, the goal is usually to minimise the total cumulative regret, which is the total sub-optimality incurred throughout training over many episodes. In contrast, the uncertain MDP setting assumes that the uncertainty is fixed: there is no opportunity to learn more about the MDP over many episodes. The sub-optimality of the final policy is evaluated in expectation over a single episode, rather than accumulated throughout training. Therefore, the uncertain MDP planning perspective is appropriate for problems where online exploration is not feasible, and the policy must be fixed offline.

Minimax regret in uncertain MDPs where only the cost function is uncertain is addressed in [91, 92, 93, 87]. For the case where the transition function is also uncertain, [82] proposes to use the sample-based representation of the uncertain MDP. The authors propose an exact solution based on solving a mixed integer linear program, however this approach does not scale. The authors also propose an approximation based on dynamic programming, although for many problems the approximation is of low-quality [83].

Thus far, all the approaches to optimising uncertain MDPs that we have discussed have been model-based. Model-free approaches to the robust uncertain MDP problem have also been proposed [94, 95, 96]. These approaches eliminate the need to specify the set of possible MDPs and instead assume that samples can be drawn from a misspecified MDP which is known to be similar to the true MDP.

Robust Adversarial Reinforcement Learning (RARL) The uncertain MDP approach of specifying a set of plausible MDPs is impractical for large-scale problems. Robust Adversarial RL (RARL) proposes an alternative approach to optimising

policies that are robust to worst-case uncertainties. RARL is posed as a two-player zero-sum game where the agent plays against an adversary which perturbs the environment [97]. RARL alternates between optimising the policy of the agent to increase the expected reward, and optimising the policy of the adversary to decrease the expected reward. In this manner, the adversary is trained to systematically find worst-case disturbances which decrease the performance of the agent policy. By training in the presence of these worst-case disturbances, the agent learns to find a policy that is robust to the presence of worst-case disturbances.

Formulations of model-free RARL differ in how they define the action space of the adversary. Options include allowing the adversary to apply perturbation forces to the simulator [98], add noise to the agent's actions [99], or periodically take over control [100].

Relevance to this thesis In Chapter 4 we consider the minimax regret criterion for uncertain MDPs. We propose a new approximation for this problem, and a means to relax the assumption that the uncertainty is independent between each state-action pair. In Chapter 6 we consider a robust MDP formulation of offline RL. To arrive at a scalable solution, we propose a model-based algorithm which is inspired by Robust Adversarial RL. Our approach treats the environment model itself as the adversary to be trained.

2.4 Bayes-Adaptive Reinforcement Learning

In the previous section, we discussed methods for finding policies that are robust to worst-case uncertainty over the MDP. An alternative approach is to instead find a policy which can *adapt* to the true underlying MDP.

The model-based Bayesian formulation of reinforcement learning [101] assumes that there is a distribution over the transition function of the underlying MDP. One way of modelling this problem is to consider the *Bayes-adaptive* MDP (BAMDP) [102]. The BAMDP augments the state space of the original MDP with the belief distribution over the underlying MDP. By integrating over the distribution of

plausible transition functions, the transition function between augmented states in the BAMDP can be computed. After observing a transition, the belief over the MDP is updated. This forms a type of belief MDP [103], and therefore the BAMDP can be viewed as a specific type of POMDP.

Acting optimally in the BAMDP results in an optimal tradeoff between exploration and exploitation: exploratory actions are only taken if they improve the expected return over the problem horizon, given the prior over MDPs. This property is referred to as Bayes-optimality. Thus, Bayes-adaptive MDPs can be thought of as bridging planning and reinforcement learning: like RL, the goal is to tradeoff exploration and exploitation to learn the best behaviour in an unknown MDP, however the BAMDP represents this as a planning problem in a known (belief) MDP [104].

Exact solutions to BAMDPs based on value iteration do not scale beyond the very smallest of problems [104]. Algorithms based on Monte Carlo tree search for POMDPs [105] have been adapted to BAMDPs [106]. These methods utilise *root sampling*, in which an MDP instantiation is sampled from the root belief at each trial. The sampled MDP is used to simulate each trial, and this approach results in the correct distribution over belief states in the BAMDP [106]. Reinforcement learning algorithms have been adapted to BAMDPs, and also utilise root sampling [107, 108].

One of the challenges of this approach is modelling the posterior distribution over MDPs. Earlier efforts considered Dirichlet distributions over transition probabilities [104], or Gaussian processes [107]. VariBAD [108] proposes to use deep latent variable models to learn to approximate the posterior distribution over MDPs, resulting in a more scalable approach.

Other approaches to the Bayesian RL problem do not attempt to generate Bayes-optimal behaviour and do not consider the BAMDP model. One line of work is based on Thompson sampling [109, 110]. In this approach, for each episode an MDP is sampled and the agent acts optimally with respect to that MDP for the entire episode. The posterior over MDPs is refined between episodes, rather than at each step per

the BAMDP approach. Another approach is to apply exploration bonuses which depend on the level of uncertainty in the model for that state-action pair [111, 112].

Relevance to this thesis In Chapter 5 we propose to optimise for risk in the BAMDP model to address both epistemic uncertainty and aleatoric uncertainty under the same framework. By avoiding risk in a BAMDP, the risk of a poor outcome occurring due to either: a) a lack of knowledge about the MDP, or b) the inherent stochasticity of the MDP, are both mitigated.

2.5 Offline Reinforcement Learning

In this section, we first provide an overview of the motivation and problem formulation of offline reinforcement learning. We then give an overview of the algorithmic challenges and existing approaches to the problem.

2.5.1 Motivation and Problem Setting

One of the major limitations of reinforcement learning is the need for extensive online exploration. For many applications, exploration can be expensive or dangerous [37], making RL impractical. For applications where online exploration is not possible, a reasonable alternative is to instead leverage large pre-existing datasets to optimise policies.

Offline (or batch) RL [113, 114] refers to the setting where we only have access to a fixed dataset. The goal is to find a performant policy using this data alone. Researchers have applied offline RL to domains such as healthcare [115, 116], natural language processing [117, 118], and robotics [38, 119, 120]

In contrast to learning from demonstration, or imitation learning [40], there is no assumption that the data is collected by an expert policy. However, unlike imitation learning, offline RL requires that the dataset is labelled with rewards to direct the learnt policy towards the desired behaviour. Therefore, offline RL is suitable for applications where the data that has been collected is noisy, and the reward function for the desired behaviour can easily be specified.

A drawback of offline RL is that the best achievable performance can be considerably worse than the optimal performance for the MDP: if the dataset does not cover high-value regions of the state space relevant to optimal behaviour, one cannot hope to obtain a performant policy [121]. A middle-ground between online and offline RL is the notion of *deployment efficient* RL [122] (also called RL with a low switching cost [123]). In this problem formulation, the number of times a new policy is a deployed for data collection is limited. This is motivated by the consideration that deploying each new policy might be a costly or risky procedure. For example, each new policy may require safety validation prior to deployment. Reducing the number of times the policy is switched lessens this burden, and makes training an RL agent in the real world more practical.

Another possibility is to use offline RL to initialise a policy, and then allow a limited amount of online exploration to fine-tune the policy [124, 125, 126]. This approach can substantially reduce the amount of online exploration required to find a near-optimal policy [124].

In the remainder of this section we focus on algorithms for the fully offline formulation of RL where no further data collection is possible. This is the most suitable approach domains where any form of random exploration in the real environment is not possible, and we do not have access to a sufficiently accurate simulator.

2.5.2 Algorithms

Offline RL poses several challenges that necessitate the design of specific algorithms. Many reinforcement learning algorithms can learn from *off-policy* data, i.e. data that was collected by a different policy. However, in practice, off-policy online RL algorithms do not perform well in the offline setting.

The main challenge of offline RL is the issue of distributional shift [114]. While the policy, value function, or model might be trained accurately under one distribution (the distribution of data in the dataset), it will be evaluated under a different distribution (the distribution generated by the output policy in the real

environment). For example, if a value function is trained to achieve low Bellman error over the dataset, this does not mean that the value function is accurate under the data distribution generated by the new policy.

This issue is exacerbated by the use of function approximation, and the fact that the policy is trained to maximise the return. Consider again the perspective of value-based methods. The Q-values for some state-action pairs will inevitably be over-estimated. The policy is trained to maximise the value function, and therefore the policy learns to systematically exploit the largest errors in the value function. In the online setting, this is not an issue as the distribution of state-action pairs generated by the policy are sampled in the environment, and this experience is used to correct inaccuracies. However, in the offline setting these inaccuracies are not corrected, and this can result in drastically over-optimistic value estimates and policies that perform poorly in the real environment [127].

For model-based approaches, which learn a dynamics model from the dataset, there is the analogous issue of *model exploitation*. The policy learns to select actions that exploit errors in the learnt model, rather than actions that are likely to perform well on the real environment, as evidenced by the dataset [128, 129, 130].

As a result of these issues, the naïve application of off-policy online RL algorithms generally does not perform well in the offline setting, and may produce erroneous actions when tested in the real environment. Earlier works on offline RL do not attempt to avoid the aforementioned issues [131, 132, 133]. More recently, the focus of offline RL research has been to develop algorithms to mitigate this issue of distributional shift. We will focus on these more recent works and give an overview of both model-free and model-based approaches.

Model-free offline RL algorithms The simplest approach to model-free offline RL is to apply an online RL algorithm, but with the additional constraint that the learnt policy is similar to the behaviour policy that generated the dataset [134, 135, 136, 118, 137, 138]. Most methods encourage the learnt policy to approximately match the distribution over actions generated by the behaviour policy. However,

this can be highly restrictive and result in suboptimal behaviour. Therefore, [127] argues for constraining the learnt policy to take actions in the *support* of the behaviour policy. However, support constraints generally require approximations to implement [139]. Therefore, subsequent works have shown that other techniques for ensuring that the policy does not leave the data distribution work better in practice [140].

Another approach is to incorporate conservatism into the value function update to prevent overestimation of the value of state-action pairs that are not present in the dataset [141, 140, 142]. For example, the value function update for CQL [140] penalises the Q-values of state-action pairs that are not contained in the dataset.

Another line of work uses uncertainty quantification to obtain more robust value estimates [143, 144, 127]. The approach proposed by [144] is inspired by clipped double Q-learning [145], which uses the minimum over two separately initialised Q-functions as the final Q-value estimate. However, [144] instead uses the minimum over a larger ensemble of many Q-networks to arrive at a pessimistic estimate of the value function in the true environment.

One-step offline RL approaches avoid the need for off-policy evaluation by only performing a single iteration of policy iteration [146, 147]. The Q-function is estimated for the behaviour policy, and policy improvement is performed under this Q-function. This approach works reasonably well in comparison to approaches which interleave policy evaluation and improvement [146]. The justification for one-step approaches is that iterative approaches propagate errors through the Bellman equation. This issue of compounding error is avoided by performing policy iteration only once.

A final class of model-free offline RL algorithms are approaches based on policy gradients with importance sampling that are tailored towards the fully offline setting [148, 149].

Model-based offline RL algorithms Model-based approaches learn a model of the environment and generate synthetic data from that model [150] to optimise a policy using either planning [151] or RL algorithms [129, 152, 153]. By training a policy on additional synthetic data, model-based approaches can potentially generalise more broadly, or be more easily adapted to new tasks [154, 152]

A simple approach to prevent model exploitation is to constrain the policy to be similar to the behaviour policy, in the same fashion as some model-free approaches [155, 122, 156]. However, like the model-free case, such constraints may prevent the algorithm from finding the best policy covered by the dataset.

Another approach is to apply reward penalties for executing state-action pairs with high uncertainty in the environment model [129, 157, 152]. If the uncertainty estimate is accurate and the reward penalties are defined appropriately, this optimises a lower bound on the value function in the true environment[129, 152]. However, in practice the uncertainty estimates for neural network models are often unreliable [158, 159, 160, 153]. COMBO [153] obviates the need for uncertainty estimation in model-based offline RL by adapting model-free techniques [140] to regularise the value function for out-of-distribution samples.

Most approaches to model-based offline RL use maximum likelihood estimates of the MDP trained using standard supervised learning [151, 122, 156, 152, 153]. However, other methods have been proposed to learn models which are more suitable for offline policy optimisation. One approach is to reweight the loss function for training the model to ensure the model is accurate under the distribution of states and actions generated by the policy [161, 162, 163]. This reduces the possibility that the policy can learn to exploit errors in the model. A *reverse* model is learnt by [164], which predicts the dynamics of the MDP backwards in time. The motivation for doing this is that it enables the generation of synthetic rollouts that always end at states which are within the dataset. This encourages the policy to learn corrective actions that stay within the distribution of the dataset. This stands in contrast to forward models which may generate trajectories that leave the dataset distribution.

Relevance to this thesis We propose two new algorithms for model-based offline RL in Chapters 6 and 7. In Chapter 6, we formulate the problem of offline RL as a Robust MDP, and propose an algorithm inspired by Robust Adversarial RL. In this instantiation of Robust Adversarial RL, we consider the adversary to be the environment model itself. The model is adversarially trained to generate pessimistic synthetic transitions for state-action pairs which are out of the dataset distribution. This is a novel approach for preventing model exploitation without using uncertainty estimation or policy constraints.

In Chapter 7 we propose to optimise for risk-aversion to solve the problem of offline RL. Risk-aversion to epistemic uncertainty solves the problem of model exploitation and distributional shift by discouraging the policy from visiting areas that are not covered by the dataset, where the epistemic uncertainty in the model is high. Our approach also achieves risk-aversion to aleatoric uncertainty under the same framework. Unlike Chapter 6, the approach in Chapter 7 requires uncertainty estimation to quantify the uncertainty in the model.

2.6 Learning from Demonstration

As discussed in the previous section, in many domains, learning a policy entirely from scratch using RL is infeasible due to the need for extensive exploration. An alternative approach is to utilise expert demonstrations to learn a policy [165]. This is referred to as either learning from demonstration, or imitation learning.

The two major paradigms of learning from demonstration are behaviour cloning, and inverse reinforcement learning [41]. On the one hand, behaviour cloning directly imitates the actions in the dataset using supervised learning [166]. On the other hand, inverse reinforcement learning finds a reward function for which the policy of the expert is optimal [167].

Even with expert demonstrations, the performance of behaviour cloning can be poor due to errors compounding as soon as the learner makes any mistake and begins to deviate from the distribution of states present in the demonstrations [168]. A policy learned via behaviour cloning does not know how to recover from such

situations. This issue of compounding error is avoided by inverse RL methods [169]. However, inverse RL approaches usually require the MDP to be solved for the current reward function in an inner loop. This may require running RL in an inner loop, so this approach can be very inefficient and still require extensive exploration of the environment [41].

To mitigate the issue of compounding error in behaviour cloning, DAGGER [170] proposes to collect more expert actions under the distribution of states generated by the current policy. This ensures that the learner sees corrective actions provided by the expert. If the reward function for the environment is known, another approach is to use learning from demonstration to initialise a policy, which is then fine-tuned using RL. Initialising a policy using demonstrations can drastically reduce the amount of exploration required for RL to find a near-optimal policy [171, 172].

In a similar vein to DAGGER, other works have proposed approaches to request expert demonstrations only if they are necessary. The "ask for help" framework introduced by [173] only asks for human input when the learner is uncertain. In [173], the agent nominally performs tabular Q-learning. If all actions in a state have a similar Q-value, the agent is deemed uncertain, and asks the human which action to take. In [174], the authors consider policies defined by a classifier. For states near a decision boundary of the classifier, a demonstration is requested. In [175], Gaussian process policies are learnt from demonstrations. A demonstration is requested if the variance of the output of the Gaussian process policy is too high. All of these approaches require a hand-tuned threshold for uncertainty.

Relevance to this thesis In Chapter 8, we consider an interactive learning from demonstration setting where expert demonstrations can be requested throughout training. Rather than developing a new algorithm for learning from demonstration, we focus on a new problem formulation. We assume that there is some cost associated with the "human effort" required to recover the system from a failed episode, as well as to provide a demonstration. Based on this cost function, the

goal is to minimise the human effort required throughout the training process. Unlike many of the aforementioned approaches, this avoids the need to hand-tune a threshold of uncertainty at which demonstrations should be requested. Instead, our approach uses the human cost objective to determine when it is worthwhile to request a demonstration.

Furthermore, we use behaviour cloning to initialise the policy prior to applying the offline RL algorithm we propose in Chapter 6. We found that initialising the policy in this manner resulted in a small performance improvement.

3

Planning for Risk-Aversion and Expected Value in MDPs

In this chapter, we consider the case where the MDP is known exactly. This is suitable for domains where there is substantial historical data that can be used to estimate the transition and reward function of the MDP, and it is safe to assume that the MDP will not change over time. We consider risk-aversion to the inherent stochasticity in the MDP, i.e. aleatoric uncertainty.

In this work, we presume that the objective is to optimise the static conditional value at risk (CVaR) of the total cost received. Static CVaR for some confidence level, α, is the mean of the worst α-portion of total costs. As discussed in Section 2.2, CVaR is commonly used in machine learning and robotics research due to the fact that it is coherent, and easy to interpret.

In this work, we focus on the tradeoff between optimising for a risk-sensitive objective and the expected performance. For most domains, risk-sensitive optimisation results in more conservative behaviour and worse performance in expectation than risk-neutral approaches. We develop an algorithm to obtain the best possible expected value subject to the constraint that CVaR remains optimal. This approach ensures that the optimal risk-sensitive behaviour is maintained, but the expected value is improved where possible as a secondary objective.

Our approach builds upon an existing planning approach for CVaR [56] that can be viewed as a two-player zero-sum game. Our algorithm first executes a policy that is optimal for CVaR. Note that CVaR is the expectation of total costs that exceed the value at risk (VaR). If, during execution, a stage is reached where there exists

a policy that is guaranteed not to exceed the VaR, then we switch to executing the policy with the best expected value that is guaranteed not to exceed the VaR. Our approach only modifies the distribution over returns that are less than the VaR, and therefore the optimal CVaR is maintained.

We evaluate this approach on several domains, including a domain based on real-world driving data, and show that we can significantly improve the expected cost obtained compared to an existing state-of-the-art algorithm, while maintaining optimal CVaR performance.

The summary in Table 1.1, which we have repeated below, shows that this chapter considers the most restrictive setting in this thesis. It assumes full knowledge of the MDP, and is only applicable to tabular MDPs. The risk-sensitive objective addressed in this chapter takes into consideration aleatoric uncertainty. However, because the MDP is assumed to be fully known, this approach does not take into consideration epistemic uncertainty.

Table 1.1: Summary of the attributes of each chapter.

	Chapter 3	**Chapter 4**	**Chapter 5**	**Chapter 6**	**Chapter 7**	**Chapter 8**
Prior knowledge	MDP fully known	MDP uncertainty set	Prior over MDPs	Fixed dataset	Fixed dataset	Expert demonstrator
Considers aleatoric uncertainty	Yes	No	Yes	No	Yes	No
Considers epistemic uncertainty	No	Yes	Yes	Yes	Yes	Yes
Risk or robustness	Risk	Robustness	Risk	Robustness	Risk	N/A
Paradigm	Tabular	Tabular	Tabular	Function Approx.	Function Approx.	Function Approx.

Marc Rigter, Paul Duckworth, Bruno Lacerda and Nick Hawes (2022). Planning for risk-aversion and expected value in MDPs. *International Conference on Automated Planning and Scheduling* (ICAPS).

Planning for Risk-Aversion and Expected Value in MDPs

Marc Rigter, Paul Duckworth, Bruno Lacerda, Nick Hawes

Oxford Robotics Institute, University of Oxford, United Kingdom
{mrigter, pduckworth, bruno, nickh }@robots.ox.ac.uk

Abstract

Planning in Markov decision processes (MDPs) typically optimises the *expected* cost. However, optimising the expectation does not consider the risk that for any given run of the MDP, the total cost received may be unacceptably high. An alternative approach is to find a policy which optimises a risk-averse objective such as conditional value at risk (CVaR). However, optimising the CVaR alone may result in poor performance in expectation. In this work, we begin by showing that there can be multiple policies which obtain the optimal CVaR. This motivates us to propose a lexicographic approach which minimises the expected cost subject to the constraint that the CVaR of the total cost is optimal. We present an algorithm for this problem and evaluate our approach on four domains. Our results demonstrate that our lexicographic approach improves the expected cost compared to the state of the art algorithm, while achieving the optimal CVaR.

Introduction

Markov decision processes (MDPs) are a common framework for decision-making under uncertainty, and have been applied to many domains such as inventory control (Ahiska et al. 2013) and robot navigation (Lacerda et al. 2019). The solution for an MDP typically optimises the expected total return, defined as either a reward to maximise or a cost to minimise. In this work, we consider the cost minimisation setting (Fig 1a). For any single run of the MDP, the total cost received is uncertain due to the MDP's stochastic transitions. In some applications we wish to compute *risk-averse* policies which prioritise avoiding the worst outcomes, rather than simply minimising the expected total cost irrespective of the variability.

As an example, consider an inventory control problem where a decision-maker decides how much stock to purchase each day, while subject to uncertain demand from customers. The policy which optimises expected value purchases large quantities of stock to maximise the expected profit. However, if demand is much lower than expected, significant losses may be incurred. A risk-averse policy purchases less stock, and is therefore less profitable on average, but avoids the possibility of large losses.

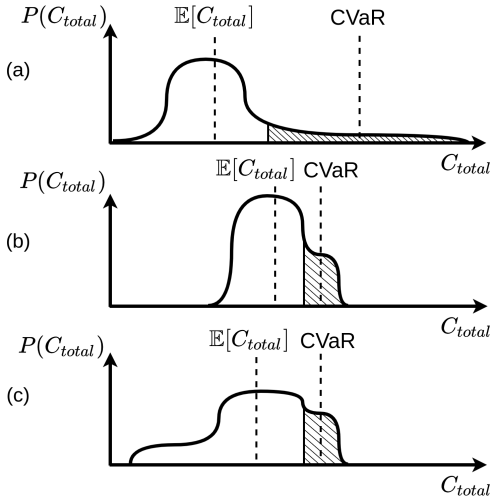


Figure 1: Illustration of distributions over the total cost, C_{total}, for different approaches: (a) expected value optimisation, (b) CVaR optimisation, and (c) our approach, which optimises the expected value subject to the constraint that CVaR is optimal. Shaded regions indicate the α-portion of the right tail of each distribution.

In this work, we focus on *conditional value at risk* (CVaR), a well known coherent risk metric (Rockafellar, Uryasev et al. 2000), in the *static* risk setting. In MDPs, the static CVaR at confidence level $\alpha \in (0, 1]$ corresponds to the mean total cost in the worst α-fraction of runs. Therefore, optimising the CVaR corresponds to optimising the α-portion of the right tail of the distribution over the total cost (Fig. 1b).

We propose that for risk-sensitive applications the *primary* objective should be to avoid the risk of a poor outcome (i.e. avoid significant losses). However, if the risk of a poor outcome has been avoided, then the *secondary* objective should be to optimise the expected value (i.e. maximise the expected profit). This motivates us to propose a lexicographic approach that optimises the expected total cost subject to the constraint that the CVaR of the total cost is optimal. During execution, the resulting policy is initially risk-averse. However, the policy may begin taking more aggressive actions to improve the expected cost, provided that there

is no longer any risk of incurring a bad run which would influence the CVaR. Fig. 1c illustrates the cost distribution for our approach, compared to optimising the expected cost or the CVaR only. As indicated by the dashed lines, our approach obtains the same CVaR as optimising for CVaR alone (Fig. 1b), but the expected cost is improved.

Our main contributions are: 1) showing that there can be multiple policies that obtain the optimal CVaR in an MDP, 2) proposing and formalising the lexicographic problem of optimising expected value in MDPs subject to the constraint of minimising CVaR, and 3) an algorithm to solve this problem, based on reducing it to a two-stage optimisation in a stochastic game. To the best of our knowledge, this is the first work to address lexicographic optimisation of CVaR and expected value in sequential decision making problems.

We evaluate our algorithm on four domains, including a road network navigation domain which uses real data from traffic sensors (Chen et al. 2001) to simulate journey times. Our experimental results demonstrate that our approach significantly improves the expected cost on all domains while attaining the optimal conditional value at risk.

Related Work

Many existing works address risk-averse optimisation in MDPs. Early work considered the expected utility framework (Howard and Matheson 1972), where the cost is transformed according to a convex utility function to achieve risk-aversion. However, it is difficult to "shape" the cost function appropriately to achieve the desired behaviour (Majumdar and Pavone 2020). Other works consider risk metrics such as the mean-variance criterion (Sobel 1982), or the value at risk (Filar, Krass, and Ross 1995). However, these risk metrics are not *coherent*, meaning that they do not satisfy properties consistent with rational decision-making. See Artzner et al. (1999) or Majumdar and Pavone (2020) for an introduction to coherent risk metrics.

Examples of coherent risk metrics include the conditional value at risk (CVaR) (Rockafellar, Uryasev et al. 2000), the entropic value at risk (Ahmadi-Javid 2012), and the Wang risk measure (Wang 2000). In this work we focus on CVaR because it is intuitive to understand, and it is the prevailing risk metric used in risk-sensitive domains such as finance (Basel Committee on Banking Supervision 2014). We consider *static* risk, where the risk metric is applied to the total cost, rather than *dynamic* risk which penalises risk at each time step (Ruszczyński 2010). For the static CVaR setting, approaches based on value iteration over an augmented state space have been proposed (Bäuerle and Ott 2011; Chow et al. 2015). Other works propose policy gradient methods to find a locally optimal solution for CVaR (Borkar and Jain 2010; Chow and Ghavamzadeh 2014; Tamar, Glassner, and Mannor 2015; Tamar et al. 2015; Prashanth 2014; Chow et al. 2017; Tang, Zhang, and Salakhutdinov 2020), or general coherent risk metrics (Tamar et al. 2016). Keramati et al. (2020) present an RL algorithm based on optimism in the face of uncertainty, while Rigter, Lacerda, and Hawes (2021) propose an approach based on Monte Carlo tree search and also consider model uncertainty.

Existing works have considered trading off the conflicting objectives of expected cost and risk. Petrik and Subramanian (2012) optimise a weighted combination of dynamic CVaR and expected value. For any given set of weightings, the performance for conditional value at risk may be sacrificed to improve the expected value. In contrast, our lexicographic approach ensures that the optimal CVaR is always attained.

An alternative approach to finding risk-averse solutions to MDPs is to impose constraints. Constrained MDPs optimise the expected value subject to a constraint on the expected value of a secondary cost function (Altman 1999; Santana, Thiébaux, and Williams 2016). By adding cost penalties to undesirable states, constraints may be used to discourage unwanted behaviour. However, this approach only constrains the expected cost and it is unclear how to choose cost penalties which result in the desired risk-averse behaviour.

CVaR-constrained problems have been addressed by existing works (Chow and Ghavamzadeh 2014; Chow et al. 2017; Borkar and Jain 2010; Prashanth 2014). In this approach, the user defines a CVaR threshold. The expected value is optimised subject to a constraint that the CVaR is less than the chosen threshold. A disadvantage of this approach is that it may be difficult to choose an appropriate CVaR threshold which is both feasible, and results in the desired level of risk-aversion.

Lexicographic approaches to multi-objective decision making in MDPs have been proposed (Mouaddib 2004; Wray, Zilberstein, and Mouaddib 2015; Lacerda, Parker, and Hawes 2015). These approaches optimise the expected value for each objective in a lexicographic ordering. Unlike these approaches, we consider risk-averse planning. To the best of our knowledge, this is the first work to propose a lexicographic approach to the optimisation of CVaR and expected value in sequential decision making problems.

Preliminaries

Conditional Value at Risk

Let Z be a bounded-mean random variable, i.e. $E[|Z|] < \infty$, on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, with cumulative distribution function $F(z) = \mathcal{P}(Z \leq z)$. In this paper, we interpret Z as the total cost which is to be minimised. The *value at risk* (VaR) at confidence level $\alpha \in (0, 1]$ is defined as $\text{VaR}_\alpha(Z) = \min\{z | F(z) \geq 1 - \alpha\}$. The *conditional value at risk* at confidence level α is defined as

$$\text{CVaR}_\alpha(Z) = \frac{1}{\alpha} \int_{1-\alpha}^{1} \text{VaR}_{1-\gamma}(Z) d\gamma. \tag{1}$$

If Z has a continuous distribution, $\text{CVaR}_\alpha(Z)$ can be defined using the more intuitive expression: $\text{CVaR}_\alpha(Z) = \text{E}[Z \mid Z \geq \text{VaR}_\alpha(Z)]$. Thus, $\text{CVaR}_\alpha(Z)$ may be interpreted as the expected value of the α-portion of the right tail of the distribution of Z. CVaR may also be defined as the expected value under a perturbed distribution using its dual representation (Rockafellar, Uryasev et al. 2000; Shapiro, Dentcheva, and Ruszczyński 2014):

$$\text{CVaR}_\alpha(Z) = \max_{\xi \in \mathcal{B}(\alpha, \mathcal{P})} \text{E}_\xi\big[Z\big], \tag{2}$$

where $\mathrm{E}_\xi[Z]$ denotes the ξ-weighted expectation of Z, and the *risk envelope*, $\mathcal{B}$, is given by

$$\mathcal{B}(\alpha, \mathcal{P}) = \left\{ \xi \mid \xi(\omega) \in \left[0, \frac{1}{\alpha}\right], \int_{\omega \in \Omega} \xi(\omega)\mathcal{P}(\omega)d\omega = 1 \right\}. \tag{3}$$

where $\mathcal{P}(\omega)$ is the probability density function if Z is continuous, and the probability mass function if Z is discrete.

Therefore, the CVaR of a random variable Z may be interpreted as the expectation of Z under a worst-case perturbed distribution, $\xi\mathcal{P}$. The risk envelope is defined so that the probability of any outcome can be increased by a factor of at most $1/\alpha$, whilst ensuring the perturbed distribution is a valid probability distribution.

Markov Decision Processes

In this work, we consider stochastic shortest path (SSP) Markov decision processes (MDPs). An SSP MDP is a tuple, $\mathcal{M} = (S, A, C, T, G, s_0)$, where S and A are finite state and action spaces; $C : S \times A \to \mathrm{R}_+$ is the cost function; $T : S \times A \times S \to [0, 1]$ is the probabilistic transition function; $G \subset S$ is the set of absorbing goal states, from which the model incurs zero cost; and s_0 is the initial state. A history of $\mathcal{M}$ is a sequence $h = s_0 a_0 s_1 a_1 \ldots$ such that $T(s_i, a_i, s_{i+1}) > 0$ for all $i \in \{0, \ldots, |h|\}$, where $|h|$ denotes the length of h. We denote the set of all finite-length histories over $\mathcal{M}$ as $\mathcal{H}^{\mathcal{M}}_{\text{fin}}$ and the set of all infinite-length histories over $\mathcal{M}$ as $\mathcal{H}^{\mathcal{M}}_{\text{inf}}$, and define the set of all histories over $\mathcal{M}$ as $\mathcal{H}^{\mathcal{M}} = \mathcal{H}^{\mathcal{M}}_{\text{fin}} \cup \mathcal{H}^{\mathcal{M}}_{\text{inf}}$. The cumulative cost function $\text{cumul}_C : \mathcal{H}^{\mathcal{M}} \to \mathrm{R}^+$ is defined such that, given history $h = s_0 a_0 s_1 a_1 \ldots$, $\text{cumul}_C(h) = \sum_{t=0}^{|h|} C(s_t, a_t)$. A history-dependent policy is a function $\pi : \mathcal{H}^{\mathcal{M}}_{\text{fin}} \to A$, and we write $\Pi^{\mathcal{M}}_{\mathcal{H}}$ to denote the set of all history-dependent policies for $\mathcal{M}$. If π only depends on the last state s_t of h, then we say π is *Markovian*, and we denote the set of all Markovian policies as $\Pi^{\mathcal{M}}$. A policy π induces a probability distribution $\mathcal{P}^{\mathcal{M}}_\pi$ over $\mathcal{H}^{\mathcal{M}}_{\text{inf}}$, and we define the cumulative cost distribution $\mathcal{C}^{\mathcal{M}}_\pi$ as the distribution over the value of cumul_C for infinite-length histories of $\mathcal{M}$ under policy π. A policy is proper at s if it reaches $s_g \in G$ from s with probability 1. A policy is proper if it is proper at all states. In an SSP MDP, the following assumptions are made (Kolobov 2012): a) there exists a proper policy, and b) every improper policy incurs infinite cost at all states where it is improper. These assumptions ensure the expected value of the cumulative cost distribution is finite for at least one policy in the SSP.

Stochastic Games

In this paper, we formulate CVaR optimisation as a turn-based two-player zero-sum stochastic shortest path game (SSPG). An SSPG between an agent and an adversary is a generalisation of an SSP MDP and can be defined using a similar tuple $\mathcal{G} = (S, A, C, T, G, s_0)$. The elements of $\mathcal{G}$ are interpreted as with MDPs, but are extended to model a two-player game. S is partitioned into a set of agent states S_{agt}, and a set of adversary states S_{adv}. Similarly, A is partitioned into a set of agent actions A_{agt}, and a set of adversary actions A_{adv}. The transition function is defined such that agent actions can only be executed in agent states, and adversary actions can only be executed in adversary states.

We denote the set of Markovian agent policies mapping agent states to agent actions as $\Pi^{\mathcal{G}}$ and the set of Markovian adversary policies, defined similarly, as $\Sigma^{\mathcal{G}}$. Similar to MDPs, a pair (π, σ) of agent-adversary policies induces a probability distribution $\mathcal{P}^{\mathcal{G}}_{(\pi,\sigma)}$ over infinite-length histories, and we define $\mathcal{C}^{\mathcal{G}}_{(\pi,\sigma)}$ as the cumulative cost distribution of $\mathcal{G}$ under π and σ. In an SSPG, the agent seeks to minimise the expected cumulative cost, whilst the adversary seeks to maximise it:

$$\min_{\pi \in \Pi^{\mathcal{G}}} \max_{\sigma \in \Sigma^{\mathcal{G}}} \mathrm{E}\big[\mathcal{C}^{\mathcal{G}}_{(\pi,\sigma)}\big]. \tag{4}$$

In an SSPG, two assumptions are made to ensure that the value in (4) is finite: a) there exists a policy for the agent which is proper for all possible policies of the adversary, and b) for any states where π and σ are improper, the expected cost for the agent is infinite (Patek and Bertsekas 1999).

CVaR Optimisation in MDPs

Many existing works have addressed the problem of optimising the CVaR of $\mathcal{C}^{\mathcal{M}}_\pi$, defined as follows.

Problem 1. Let $\mathcal{M}$ be an MDP. Find the optimal CVaR of the cumulative cost at confidence level α:

$$\min_{\pi \in \Pi^{\mathcal{M}}_{\mathcal{H}}} \text{CVaR}_\alpha(\mathcal{C}^{\mathcal{M}}_\pi). \tag{5}$$

Note that the optimal policy for Problem 1 may be history-dependent (Bäuerle and Ott 2011). Methods based on dynamic programming have been proposed to solve Problem 1 (Bäuerle and Ott 2011; Chow et al. 2015; Pflug and Pichler 2016). In particular, the current state of the art approach (Chow et al. 2015) formulates Problem 1 as the expected value in an SSPG against an adversary which modifies the transition probabilities. This formulation is based on the CVaR representation theorem in Eq. 2, where CVaR may be represented as the expected value under a perturbed probability distribution. The SSPG is defined so that the ability for the adversary to perturb the transition probabilities corresponds to the risk envelope given in Eq. 3.

Formally, the CVaR SSPG is defined by the tuple $\mathcal{G}^+ = (S^+, A^+, C^+, T^+, G^+, s_0^+)$. The state space $S^+ = S \times [0, 1] \times (A \cup \{\perp\})$ is the original MDP state space augmented with a continuous state factor, $y \in [0, 1]$, representing the "budget" of the adversary to perturb the probabilities; and a state factor $a \in A \cup \{\perp\}$ indicating the most recent agent action if it is the adversary's turn to choose an action, and $\perp$ if is the agent's turn. The action space is defined as $A^+ = A \cup \Xi$, i.e. the agent actions are the actions in the original MDP, and the adversary actions are a set Ξ that will be defined next.

The CVaR SSPG transition dynamics are as follows. Given an agent state, $(s, y, \perp) \in S^+_{agt}$, the agent applies an action, $a \in A$, and receives cost $C^+((s, y, \perp), a) = C(s, a)$. The state then transitions to the adversary state $(s, y, a) \in S^+_{adv}$. The adversary then chooses an action to perturb the

original MDP transition probabilities from a continuous action space defined as:

$$\Xi(s,y,a) = \Big\{\xi \in \mathrm{R}^{|S|} \;\Big|\; 0 \leq \xi(s') \leq \eta \;\; \forall s' \text{ and } \sum_{s' \in S} \big[\xi(s') \cdot T(s,a,s')\big] = 1\Big\}, \quad (6)$$

where $\eta = \infty$ if $y = 0$, and $\eta = 1/y$ otherwise; T is the original MDP transition function. Eq. 6 restricts the adversary actions so the probability of any history is increased by at most $1/y$, and the perturbed transition probabilities remain a valid probability distribution. After the adversary chooses the perturbation action $\xi \in \Xi(s,y,a)$, the game transitions back to an agent state $(s', y \cdot \xi(s'), \perp) \in S^+_{agt}$ according to the following transition function where the probability of each successor s' in the original MDP is perturbed by the factor $\xi(s')$:

$$T^+((s,y,a),\xi,(s',y\xi(s'),\perp)) = \xi(s')T(s,a,s'). \quad (7)$$

Finally, we define the initial augmented state as $s^+_0 = (s_0, \alpha, \perp) \in S^+_{agt}$, where the y state variable is set to the CVaR confidence level α. We also define G^+ as the set of goal states on the augmented state space corresponding to goal states in the original MDP.

Chow et al. (2015) showed that the minimax expected value equilibrium for $\mathcal{G}^+$ corresponds to the optimal CVaR.

Proposition 1. *(Chow et al. 2015) Let $\mathcal{G}^+$ be a CVaR SSPG corresponding to original MDP $\mathcal{M}$. Then:*

$$\min_{\pi \in \Pi^{\mathcal{M}}_{\mathcal{H}}} \mathrm{CVaR}_\alpha(\mathcal{C}^{\mathcal{M}}_\pi) = \min_{\pi \in \Pi^{\mathcal{G}^+}} \max_{\sigma \in \Sigma^{\mathcal{G}^+}} \mathrm{E}\big[\mathcal{C}^{\mathcal{G}^+}_{(\pi,\sigma)}\big]. \quad (8)$$

Proposition 1 holds because the y state variable keeps track of the total multiplicative perturbation to the probability of any history. Thus, the constraints in Eq. 6 ensure that the maximum perturbation to the probability of any history from the initial state is $1/\alpha$, and that the perturbed distribution over histories is a valid probability distribution. Therefore, the admissible adversarial perturbation actions in $\mathcal{G}^+$ correspond to the risk envelope in Eq. 3. According to Eq. 2, the CVaR is the expected value under the perturbations in the risk-envelope that maximise the expected cost.

To compute the solution to Eq. 8, we denote the value function for the augmented state by $V_{\mathrm{CV}}(s,y,\perp) = \min_{\pi \in \Pi^{\mathcal{M}}_{\mathcal{H}}} \mathrm{CVaR}_y(\mathcal{C}^{\mathcal{M}}_\pi)$. This value function can be computed using minimax value iteration over $\mathcal{G}^+$ using the following Bellman equation:

$$V_{\mathrm{CV}}(s,y,\perp) = \min_{a \in A} \Bigg[C(s,a) + \max_{\xi \in \Xi(s,y,a)} \sum_{s' \in S} \xi(s')T(s,a,s')V_{\mathrm{CV}}(s', y\xi(s'), \perp)\Bigg]. \quad (9)$$

We denote the policy corresponding to the value function obtained by solving Eq. 9 by π_{CV}. As the augmented state space contains a continuous variable and the action space is also continuous, Chow et al. (2015) use approximate dynamic programming with linear function approximation.

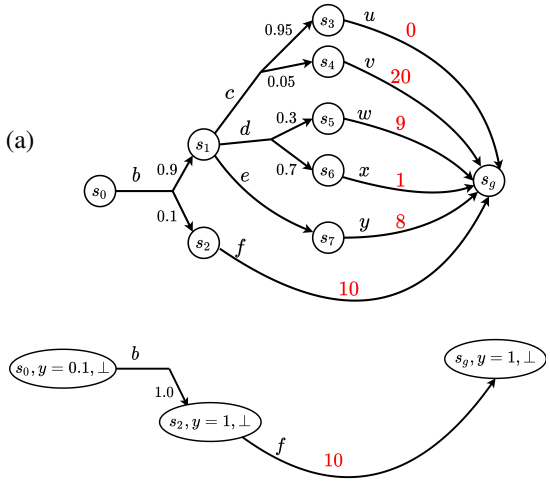


Figure 2: (a) Example Markov decision process with initial state s_0 and goal state s_g. Letters indicate actions. Costs are assumed to be zero unless indicated otherwise in red. (b) Corresponding Markov chain induced by the solution to Equation 8 when $\alpha = 0.1$.

Example 1. In Fig. 2 we present an illustrative example of the approach to CVaR optimisation from Chow et al. (2015). Fig. 2a depicts an example MDP. Fig. 2b shows the solution to Eq. 8 to optimise the CVaR when $\alpha = 0.1$, illustrated as a Markov chain (MC): the adversary perturbs the transition probabilities so that the probability of transitioning from s_0 to s_2 is increased from 0.1 to 1 and, conversely, the probability of transitioning from s_0 to s_1 is decreased from 0.9 to 0. The expected value in this MC is 10, and by Proposition 1 this is the optimal CVaR at $\alpha = 0.1$ in the original MDP.

Remark 1. State s_1 is reachable in the original MDP, but assigned zero probability by the adversary. At states assigned zero probability by the adversary, $y = 0$ in the CVaR SSPG. When $y = 0$, the adversary has unlimited power to perturb the transition probabilities (Eq. 6). Therefore, when $y = 0$ the minimax value iteration in Eq. 9 optimises for the minimum worst-case cost, as the adversary has the power to make the agent transition deterministically to the worst possible successor state for all future transitions. We will refer to the approach from Chow et al. (2015) as *CVaR-Worst-Case* because of this property that for histories assigned zero probability by the adversary, this method optimises for the minimum worst-case total cost. In the following section we show that this behaviour is unnecessarily conservative to optimise CVaR. We propose a method for finding an alternative policy to execute in such situations that optimises the expected value while maintaining the optimal CVaR.

Lexicographic Optimisation of CVaR

In this section we present the contributions of this paper. We begin with a formal problem statement.

Problem 2. Let $\mathcal{M}$ be an MDP. Find the policy π^* that optimises the expected cost subject to the constraint that π^*

obtains the optimal CVaR at confidence level α:

$$\begin{aligned}\pi^* = &\arg\min_{\pi \in \Pi^{\mathcal{M}}_{\mathcal{H}}} \mathrm{E}\big[\mathcal{C}^{\mathcal{M}}_{\pi}\big], \\ &\text{s.t.} \quad \mathrm{CVaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi}) = \min_{\pi' \in \Pi^{\mathcal{M}}_{\mathcal{H}}} \mathrm{CVaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi'}).\end{aligned} \tag{10}$$

Our approach to this problem extends the approach to CVaR optimisation from Chow et al. (2015), which we have outlined. We begin by emphasising that in the approach in Chow et al. (2015), some histories in the original MDP have zero probability under the adversarial perturbations obtained by solving Eq. 8. For example, all histories passing through s_1 in Fig. 2a are not reachable under the adversarial perturbations, as illustrated in Fig. 2b. Intuitively, this suggests that such histories do not contribute to the CVaR, as the CVaR can be computed as the expected value under the perturbed transition probabilities (Proposition 1).

In this paper, we investigate finding an alternative policy to execute when we reach such histories which are assigned zero probability by the adversary in the CVaR SSPG. By finding an appropriate alternative policy to execute in these situations, we are able to optimise the expected value whilst maintaining the optimal CVaR. We now state two properties of these histories which will be useful to ensure the policies obtained by our algorithm maintain the optimal CVaR. Full proofs are in the supplementary material.

Proposition 2. *Let π denote a CVaR-optimal policy for confidence level α, and let σ denote the corresponding adversary policy from Eq. 8. For some history, h, if $\mathcal{P}^{\mathcal{M}}_{\pi}(h) > 0$ and $\mathcal{P}^{\mathcal{G}}_{(\pi,\sigma)}(h) = 0$, there exists a policy $\pi' \in \Pi^{\mathcal{M}}_{\mathcal{H}}$ which may be executed from h onwards for which the total cost received over the run is guaranteed to be less than or equal to $\mathit{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$.*

Proof Sketch: We prove Proposition 2 by showing that the original policy, π, satisfies this condition. Denote by $\mathcal{H}^{\mathcal{M}}_{g}$ the set all of histories ending at a goal state, and let $h_g \in \mathcal{H}^{\mathcal{M}}_{g}$ denote any such history. Using the CVaR representation theorem in Equation 2, we can show that if h_g has non-zero probability under π in the original MDP, but is assigned zero probability by the adversary in the CVaR SSPG, then the total cost of h_g must be less than or equal to $\mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$:

$$\mathcal{P}^{\mathcal{M}}_{\pi}(h_g) > 0 \text{ and } \mathcal{P}^{\mathcal{G}}_{(\pi,\sigma)}(h_g) = 0 \implies \mathrm{cumul}_C(h_g) \leq \mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi}). \tag{11}$$

Now consider any history, h, which has not reached the goal. Assume that $\mathcal{P}^{\mathcal{M}}_{\pi}(h) > 0$ and $\mathcal{P}^{\mathcal{G}}_{(\pi,\sigma)}(h) = 0$. For all histories $h_g \in \mathcal{H}^{\mathcal{M}}_{g}$ reachable after h under π (i.e. for which $\mathcal{P}^{\mathcal{M}}_{\pi}(h_g) > 0$), we have that $\mathcal{P}^{\mathcal{G}}_{(\pi,\sigma)}(h_g) = 0$. Therefore, by Equation 11 all histories h_g reachable after h under π are guaranteed to have $\mathrm{cumul}_C(h_g) \leq \mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$. This proves that by continuing to execute π, the total cost received over the run is guaranteed to be less than or equal to $\mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$.

Proposition 2 shows that if a state is reached which is assigned zero probability by the adversary, then there must exist a policy which can be executed from that state onwards for which the total cost is guaranteed not to exceed the VaR. The following proposition states that by switching to any policy that guarantees that the cost will not exceed the VaR, the optimal CVaR is maintained.

Proposition 3. *During execution of policy π optimal for $\mathrm{CVaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$, we may switch to any policy π' and still attain the same CVaR, provided that π' is guaranteed to incur total cost of less than or equal to $\mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$.*

Proof Sketch: $\mathrm{CVaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$ is computed by integrating over the distribution of costs greater than or equal to $\mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$ (Equation 1). Because switching to π' does not result in any costs greater than $\mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi})$, the strategy of switching to π' cannot increase the CVaR. Switching to π' cannot decrease the CVaR as π already attains the optimal CVaR. Therefore, the strategy of switching to an appropriate π' must attain the same CVaR.

CVaR-Expected-Value

Proposition 3 establishes that during execution of π_{CV} we can switch to another policy, π', without influencing the CVaR, provided that the total cost of executing π' is guaranteed not to exceed $\mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi_{\mathrm{CV}}})$. Proposition 2 establishes that such a policy exists if the current history is assigned zero probability by the adversary in the CVaR SSPG. However, as we shall illustrate in the following example there may be multiple policies which satisfy this criterion. Of these policies, we would like to find the one which optimises our secondary objective of expected value as stated in Problem 2.

Example 2. Consider again the model in Fig. 2. We established in Example 1 that the optimal CVaR at $\alpha = 0.1$ is 10, which can be computed as the expected value under the Markov chain in Fig. 2b. The corresponding VaR at $\alpha = 0.1$ is also 10. All of the histories passing through s_1 are assigned zero probability by the adversarial perturbations in the CVaR SSPG. At s_1, we can continue to execute any policy for which all histories are guaranteed to reach the goal with total cost less than or equal to the VaR threshold of 10. Executing either d or e at s_1 satisfies this property, and maintains the optimal CVaR of 10. The approach of Chow et al. (2015), which we refer to as *CVaR-Worst-Case*, would choose action e as in this situation it optimises for the minimum worst-case cost (see Remark 1). However, d achieves better expected value and still attains the optimal CVaR of 10. Therefore, to solve Problem 2, the policy should choose d. On the other hand, executing c attains a sub-optimal CVaR of greater than 10, as some histories would receive a total cost of 20.

We wish to develop a general approach to finding the policy, π', which optimises the expected value, subject to the constraint that the worst-case total cost must not exceed $\mathrm{VaR}_{\alpha}(\mathcal{C}^{\mathcal{M}}_{\pi_{\mathrm{CV}}})$. We know that such a policy exists for histories assigned zero probability by the adversary in the CVaR SSPG. Therefore, by switching from π_{CV} to π' when such histories occur, we can optimise the expected value while still attaining the optimal CVaR, thus solving Problem 2.

We first compute the minimum worst-case total cost at each state, $V_{worst}(s)$, using the following minimax Bellman

equation which assumes that the agent always transitions deterministically to the worst-case successor state:

$$V_{worst}(s) = \min_{a \in A} \Big[C(s,a) + \max_{s' \in S} \Big(\mathbf{1}(T(s,a,s') > 0) \cdot V_{worst}(s') \Big) \Big], \tag{12}$$

where $\mathbf{1}$ denotes the indicator function. Let π_{worst} denote the policy corresponding to V_{worst}. We also compute the optimal worst-case Q-values:

$$Q_{worst}(s,a) = C(s,a) + \max_{s' \in S} \mathbf{1}(T(s,a,s') > 0) \cdot V_{worst}(s'). \tag{13}$$

Now, assume that the history so far is h, and that the cost so far in the history is $\text{cumul}_C(h)$. To maintain the optimal CVaR according to Proposition 3, we can allow an action a to be executed at h if

$$\text{cumul}_C(h) + Q^*_{worst}(s,a) \leq \text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}}). \tag{14}$$

For any action a which satisfies Eq. 14, after taking a we can then execute π_{worst} and guarantee that the total cost will not exceed $\text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})$. Thus, by finding a policy which optimises the expected value subject to the constraint that actions must satisfy Eq. 14, we can find the policy with best expected value which is guaranteed to have total cost less than or equal to $\text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})$ and maintain the optimal CVaR.

Note that which actions are allowed in Eq. 14 depends not only on the current state, but also on the cost received so far. To arrive at an offline solution we create an augmented MDP, $\mathcal{M}' = (S', A, T', C', G', s'_0)$, where we restrict which actions can be executed depending on the cost received so far. We start by augmenting the state space of the original MDP such that $S' = S \times [0, \text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})]$. This augmented state keeps track of how much cost has been incurred. The action set A is the same as the original MDP. The transition function ensures that actions are only enabled if they satisfy Eq. 14, and propagates the accumulated cost appropriately:

$$T'((s,c),a,(s',c')) = \begin{cases} T(s,a,s') & \text{if } c + Q_{worst}(s,a) \leq \text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}}) \\ & \text{and } c' = c + C(s,a), \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

The cost function is $C'((s,c),a) = C(s,a)$; the goal states are $G' = G \times [0, \text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})]$; and the initial state is $s'_0 = (s_0, 0)$. The value function corresponding to the following standard MDP Bellman equation is the optimal expected value subject to the constraint that $\text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})$ will not be exceeded for any possible history.

$$V^{\mathcal{M}'}(s,c) = \min_{a \in A} \left[C'(s,a) + \sum_{(s',c')} T'\big((s,c),a,(s',c')\big) \cdot V^{\mathcal{M}'}(s',c') \right]. \tag{16}$$

Solving Eq. 16 is not straightforward due to the additional continuous state variable which prohibits standard discrete value iteration. Therefore, we instead apply value iteration with linear interpolation (Bertsekas 2007) for the continuous variable in the same manner as (Chow et al. 2015).

Algorithm 1: *CVaR-Expected-Value*

get π_{CV} and $V_{\text{CV}}(s,y)$ by solving Equation 9
get π' by solving Equation 16
$s^+ \leftarrow s_0^+$
$c \leftarrow 0$
function `ExecuteEpisode()`
  do
    $a \leftarrow \pi_{\text{CV}}(s^+)$
    $\xi = \arg\max_{\xi \in \Xi(s,y,a)} \sum_{s' \in S} \xi(s') \cdot T(s,a,s') \cdot V_{\text{CV}}(s', y \cdot \xi(s'), \perp)$
    $s^+ \leftarrow (s', y \cdot \xi(s'), \perp)$, where s' is successor after executing a in the MDP
    $c \leftarrow c + C(s,a)$
    if $s^+ \in G^+$ **:**
      return
  while $\xi(s') > 0$

  while $s \notin G$ **:**
    $a \leftarrow \pi'(s,c)$
    $s \leftarrow s'$, where s' is successor after executing a in the MDP
    $c \leftarrow c + C(s,a)$
  return

We write π' to denote the optimal policy corresponding to the value function $V^{\mathcal{M}'}$. By first executing π_{CV}, and then executing π' on histories assigned zero probability by the adversary in the CVaR SSPG, we switch to using the policy with best expected value subject to the constraint that the optimal CVaR is maintained. Therefore, this approach solves Problem 2. This approach, which we refer to as *CVaR-Expected-Value* is described in Algorithm 1.

In this approach, we decide which actions are pruned out based on $\text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})$. To estimate $\text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})$, one approach would be to compute the worst-possible history in the Markov chain induced by Eq. 8. However, this is computationally challenging as the number of reachable histories in the continuous state space may be very large. Therefore, we take the simple approach of first executing π_{CV} (*CVaR-Worst-Case*) for many episodes and compute a Monte Carlo estimate for $\text{VaR}_\alpha(\mathcal{C}^{\mathcal{M}}_{\pi_{\text{CV}}})$ (Hong, Hu, and Liu 2014), which we then use to restrict actions according to Eq. 15.

Experiments

The code and data used to run the experiments is included in the supplementary material and will be made publicly available. We experimentally evaluate the following three approaches: *CVaR-Worst-Case* (*CVaR-WC*), *CVaR-Expected-Value* (*CVaR-EV*), and *Expected Value* (*EV*). *Expected Value* is the policy which optimises the expected value only. All algorithms are implemented in C++ and Gurobi is used to solve linear programs where necessary. We use 30 interpolation points for α to solve Eq. 9, and 100 interpolation points for the cumulative cost to solve Eq. 16. The experiments used a 3.2 GHz Intel i7 processor with 64 GB of RAM.

We compare the approaches on the following four do-

mains. The three synthetic domains are from the literature, and the fourth domain we introduce is based on real data.

Inventory Control (IC) We consider the stochastic inventory control problem (Puterman 2005; Ahmed et al. 2017), with 10 decision stages. The current number of units in the inventory is n, and the maximum number of units is $N = 20$. The action set is $a \in \{0, \dots, N - n\}$, representing the amount of stock to purchase at each stage. There is an expense of $p_u = 1$ for purchasing each item of stock. Items are sold according to the stochastic demand for each stage, $d \in \{0, \dots, N\}$. A revenue of $r = 3$ is received for each unit sold. For any items of stock which do not sell, there is a holding expense at each stage of $p_h = 1$ per unit. Thus, the profit received at each stage is

$$min\{d, n + a\} \cdot r - a \cdot p_u - max\{n + a - d, 0\} \cdot p_h$$

We assume that the demand is modelled as random walk. At each stage, $d = d_{prev} + \Delta d$, where d_{prev} is the demand at the previous stage, and Δd is uniformly distributed between ± 5. At the first stage, $d_{prev} = 10$. The demand at the previous stage is included in the state space.

We model all domains as SSP MDPs, where a positive cost function is minimised. To convert rewards to a positive cost function, we take the standard approach (Kolobov 2012) of defining the cost function to be the maximum possible reward minus the reward received. In Inventory Control, the maximum possible profit is: $\max(\text{profit}) = 400$. The cost for an episode is $\max(\text{profit})$ minus the cumulative profit received over all stages. Therefore, a cost of under 400 represents a net profit, and a cost over 400 represents a net loss.

Betting Game (BG) We adapt this domain from the literature on CVaR in MDPs (Bäuerle and Ott 2011). The state is represented by two factors: $(money, stage)$. The agent begins with $money = 5$. The amount of money that the agent can have is limited between 0, and $\max(money) = 100$. At each stage the agent may choose to place a bet from $bets = \{0, 1, 2, 3, 4, 5\}$ provided that sufficient money is available. If the agent wins the game at that stage, an amount of money equal to the bet placed is received. If the agent loses the stage, the money bet is lost. If the agent wins the jackpot, the agent receives $10\times$ the amount bet. For each stage, the probability of winning is 0.7, the probability of winning the jackpot is 0.05, and the probability of losing is 0.25. After 10 stages, the cost received is $\max(money) - money$.

Deep Sea Treasure (DST) This domain is adapted from the literature on multi-objective optimisation in MDPs (Vamplew et al. 2011). A submarine navigates a grid-world to collect one of many treasures, each of which is associated with a reward value, r (illustrated in the supplementary material). At each timestep, the agent chooses from 8 actions corresponding to directions of travel and moves to the corresponding square with probability 0.6 and each of the adjacent squares with probability 0.2. The episode ends when the agent reaches a treasure or the horizon of 15 steps is reached. The agent incurs a cost of 5 at each timestep, and a terminal cost of $500 - r$. There is a tradeoff between risk and expected cost as if the submarine travels further it may collect more valuable treasure, and therefore incur less cost, but risks running out of time before reaching any treasure.

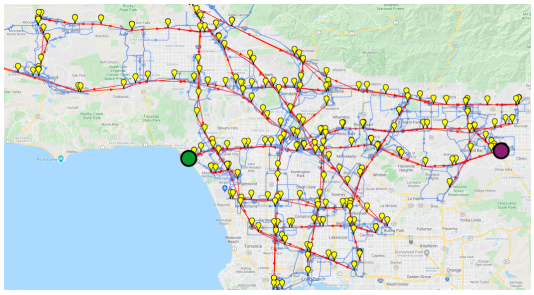


Figure 3: Autonomous Vehicle Navigation domain: MDP state space (yellow balloons) and transitions (red and blue edges) overlaid onto map of Los Angeles, USA. The green and purples circles indicate the start and goal locations.

Autonomous Navigation (AN) An autonomous vehicle must plan routes across Los Angeles, USA between a start and goal location as illustrated in Fig. 3. We access real road traffic data collected by Caltrans Performance Measurement System (PeMS) by over 39,000 real-time traffic sensors deployed across the major metropolitan areas of California (Chen et al. 2001). We select a subset of 263 sensors along the major freeways, shown as yellow markers in Figure 3. We specify two types of transitions: i) *freeway* transitions (red) along a specified freeway where the transition time distribution is generated from historical PeMS traffic data (discrete with 10 bins), and ii) *between-freeway* transitions (blue) where each state is connected to its three nearest neighbours on other freeways and the transition time is normally distributed around the expected duration obtained from querying the Google Routes API. In this domain, the cost is the journey duration in minutes. To simulate rare traffic jams on the freeways, uniform noise in [0, 0.1] is added to the slowest *freeway* transitions (and probabilities renormalised). This is motivated by knowledge that rare but severe traffic can affect transition durations on freeways, and introduces a tradeoff between risk and expected cost.

Results

The results of our experiments are in Table 1. The rows in the table indicate the method and the confidence level that the method is set to optimise. The columns indicate the performance of the policy for each objective over 20,000 evaluation episodes in each domain. We measure performance for $\text{CVaR}_{0.02}$ (i.e. the mean cost of the worst 2% of runs), $\text{CVaR}_{0.2}$ (worst 20%), and the expected value. We expect the *CVaR-EV* method to match the *CVaR-WC* performance on its CVaR measurement column. We also expect *CVaR-EV* to achieve lower cost than *CVaR-WC* in expectation.

In the Inventory Control domain, *EV* performs the best for expected value, but the worst for the CVaR objectives. For the $\text{CVaR}_{0.02}$ objective, *EV* obtains 416, representing an average net loss of 16 on the worst 2% of runs. In contrast, when optimising for $\text{CVaR}_{0.02}$ (i.e. $\alpha = 0.02$), both *CVaR-WC* and *CVaR-EV* obtain values of 386 (profit of 14) for the $\text{CVaR}_{0.02}$ objective. This illustrates that the risk-averse approaches avoid the possibility of losses on poor runs. Similarly, we see that *CVaR-EV* and *CVaR-WC* equally out-

	Inventory Control (IC)		Betting Game (BG)		Deep Sea Treasure (DST)		Autonomous Navigation (AN)	
Method	$\text{CVaR}_{0.02}$	Expected Value	$\text{CVaR}_{0.02}$	Expected Value	$\text{CVaR}_{0.02}$	Expected Value	$\text{CVaR}_{0.02}$	Expected Value
CVaR-WC ($\alpha=0.02$)	386.49 (0.23)	286.18 (0.50)	**95.0** (0.0)	**95.0** (0.0)	502.25 (0.78)	402.74 (0.43)	210.20 (1.36)	167.35 (0.11))
CVaR-EV ($\alpha=0.02$)	**386.92** (0.24)	**250.38** (0.66)	**95.0** (0.0)	**95.0** (0.0)	**502.98** (0.83)	**352.06** (0.57)	**211.16** (1.40)	**164.14** (0.12)
Expected Value	416.42 (0.60)	235.62 (0.70)	100.0 (0.0)	58.26 (0.22)	575.0 (0.0)	315.15 (0.66)	315.99 (2.12)	120.19 (0.38)
Method	$\text{CVaR}_{0.2}$	Expected Value	$\text{CVaR}_{0.2}$	Expected Value	$\text{CVaR}_{0.2}$	Expected Value	$\text{CVaR}_{0.2}$	Expected Value
CVaR-WC ($\alpha=0.2$)	360.65 (0.31)	272.51 (0.48)	91.97 (0.08)	82.95 (0.06)	422.29 (0.62)	349.45 (0.42)	172.05 (0.80)	134.12 (0.21)
CVaR-EV ($\alpha=0.2$)	**360.29** (0.31)	**250.08** (0.63)	**91.86** (0.08)	**75.63** (0.16)	**422.80** (0.63)	**340.12** (0.49)	**171.58** (0.80)	**132.77** (0.22)
Expected Value	370.54 (0.37)	235.62 (0.70)	97.36 (0.07)	58.26 (0.22)	453.07 (1.09)	315.15 (0.66)	217.86 (0.71)	120.19 (0.38)

Table 1: Results from evaluating each method for 20,000 episodes. Rows indicate the optimisation method. Columns indicate the performance for the CVaR and expected value objectives in each domain. The bolded results indicate the method with best performance for expected value given that the CVaR objective is optimal (i.e. Problem 2). Brackets indicate standard errors.

perform *EV* at optimising the $\text{CVaR}_{0.2}$ objective. For both $\alpha = 0.02$ and $\alpha = 0.2$, *CVaR-EV* significantly improves the expected value compared to *CVaR-WC*, meaning that the average profitability is improved while still avoiding risks.

Histograms for the total costs received by *CVaR-WC* ($\alpha = 0.02$), *CVaR-EV* ($\alpha = 0.02$), and *EV* are shown in Figure 4 for the Inventory Control domain. Equivalent plots for the other domains are in the supplementary material. We see that while *EV* performs the best in expectation, it incurs the most runs where the cost is above 400, corresponding to net losses. Therefore, *EV* performs worse at $\text{CVaR}_{0.02}$. For *CVaR-WC* and *CVaR-EV*, the right tail of the distributions are equivalent, resulting in the same performance for $\text{CVaR}_{0.02}$. However, for *CVaR-EV*, the left side of the distribution is spread further left, improving the expected value.

Similarly, in the Betting Game domain *EV* performs the best for expected value, but worse for the CVaR objectives. For *CVaR-WC* and *CVaR-EV* with $\alpha = 0.02$, the optimal policy is never to bet, and this policy attains the best performance for the $\text{CVaR}_{0.02}$ objective. For $\alpha = 0.2$, both *CVaR-WC* and *CVaR-EV* achieve similar performance for $\text{CVaR}_{0.2}$. However, *CVaR-EV* attains significantly lower cost in expectation. This occurs because winning the jackpot is usually sufficient to guarantee that the VaR will not be exceeded. In these situations, *CVaR-WC* stops betting. On the other hand, *CVaR-EV* bets aggressively in these situations, as bets can safely be made without the risk of having a bad run which would influence the CVaR.

For both the Deep Sea Treasure and Autonomous Navigation domains, we make the same observation that both *CVaR-WC* and *CVaR-EV* achieve the same CVaR performance when optimising each of the $\text{CVaR}_{0.02}$ and $\text{CVaR}_{0.2}$ objectives. However, *CVaR-EV* obtains better expected value performance. In both domains, we also see that *EV* performs the best in expectation, but less well at the CVaR objectives.

The computation times in Table 2 indicate that for three out of four domains, the computation required for *CVaR-EV* is only a moderate (5%-30%) increase over the computation required for *CVaR-WC*. For Inventory Control, the increase in computation time is more substantial (150%).

Conclusion

In this paper, we have presented an approach to optimising the expected value in MDPs subject to the constraint that the CVaR is optimal. Our experimental evaluation on four domains has demonstrated that our approach is able to attain optimal CVaR while improving the expected performance compared to the current state of the art method. In future work, we wish to improve scalability by extending our approach to use labelled real-time dynamic programming (Bonet and Geffner 2003) rather than value iteration over the entire state space.

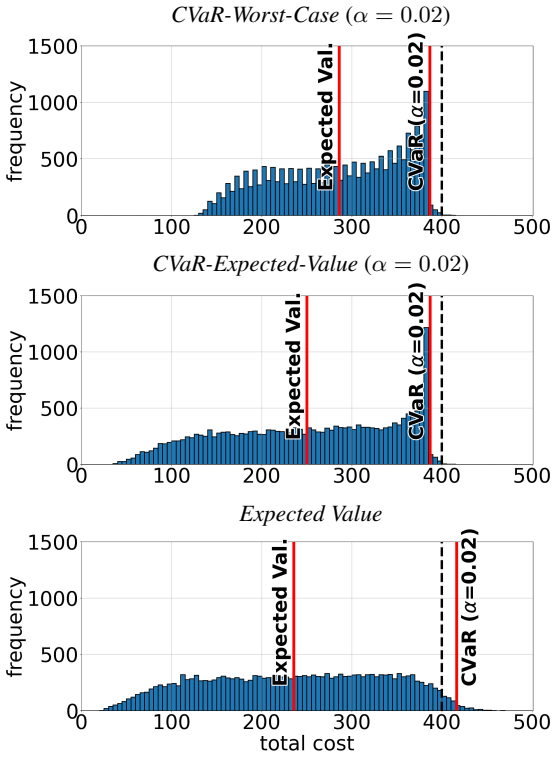


Figure 4: Histograms for the total cost received over 20000 evaluation episodes in the Inventory Control domain. Total costs of greater than 400 (dashed black line) represent losses, whereas total costs of less than 400 represent profit.

Method	IC	BG	DST	AN
CVaR-Worst-Case	19637	6215	8156	18327
CVaR-Expected-Value	48527	6526	10653	23020
Expected Value	8303	34.5	505	1876

Table 2: Computation times for each approach in seconds.

References

Ahiska, S. S.; Appaji, S. R.; King, R. E.; and Warsing Jr, D. P. 2013. A Markov decision process-based policy characterization approach for a stochastic inventory control problem with unreliable sourcing. *International Journal of Production Economics*, 144(2): 485–496.

Ahmadi-Javid, A. 2012. Entropic value-at-risk: A new coherent risk measure. *Journal of Optimization Theory and Applications*, 155(3): 1105–1123.

Ahmed, A.; Varakantham, P.; Lowalekar, M.; Adulyasak, Y.; and Jaillet, P. 2017. Sampling based approaches for minimizing regret in uncertain Markov decision processes (MDPs). *Journal of Artificial Intelligence Research*, 59: 229–264.

Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

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

Basel Committee on Banking Supervision. 2014. Fundamental review of the trading book: A revised market risk framework. *Bank for International Settlements*.

Bäuerle, N.; and Ott, J. 2011. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3): 361–379.

Bertsekas, D. P. 2007. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition.

Bonet, B.; and Geffner, H. 2003. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *International Conference on Automated Planning and Scheduling*, 12–21.

Borkar, V.; and Jain, R. 2010. Risk-constrained Markov decision processes. In *49th IEEE Conference on Decision and Control (CDC)*, 2664–2669. IEEE.

Chen, C.; Petty, K.; Skabardonis, A.; Varaiya, P.; and Jia, Z. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 1748(1): 96–102.

Chow, Y.; and Ghavamzadeh, M. 2014. Algorithms for CVaR optimization in MDPs. In *Advances in Neural Information Processing Systems*, 3509–3517.

Chow, Y.; Ghavamzadeh, M.; Janson, L.; and Pavone, M. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1): 6070–6120.

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*, 1522–1530.

Filar, J. A.; Krass, D.; and Ross, K. W. 1995. Percentile performance criteria for limiting average Markov decision processes. *IEEE Transactions on Automatic Control*, 40(1): 2–10.

Hong, L. J.; Hu, Z.; and Liu, G. 2014. Monte Carlo methods for value-at-risk and conditional value-at-risk: a review. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 1–37.

Howard, R. A.; and Matheson, J. E. 1972. Risk-sensitive Markov decision processes. *Management science*, 18(7): 356–369.

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

Kolobov, A. 2012. Planning with Markov decision processes: An AI perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1): 1–210.

Lacerda, B.; Faruq, F.; Parker, D.; and Hawes, N. 2019. Probabilistic planning with formal performance guarantees for mobile service robots. *The International Journal of Robotics Research*, 38(9): 1098–1123.

Lacerda, B.; Parker, D.; and Hawes, N. 2015. Nested Value Iteration for Partially Satisfiable Co-Safe LTL Specifications. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (SDMIA)*.

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

Mouaddib, A.-I. 2004. Multi-objective decision-theoretic path planning. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings.*, volume 3, 2814–2819.

Patek, S. D.; and Bertsekas, D. P. 1999. Stochastic shortest path games. *SIAM Journal on Control and Optimization*, 804–824.

Petrik, M.; and Subramanian, D. 2012. An Approximate Solution Method for Large Risk-Averse Markov Decision Processes. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 805–814.

Pflug, G. C.; and Pichler, A. 2016. Time-consistent decisions and temporal decomposition of coherent risk functionals. *Mathematics of Operations Research*, 41(2): 682–699.

Prashanth, L. 2014. Policy gradients for CVaR-constrained MDPs. In *International Conference on Algorithmic Learning Theory*, 155–169. Springer.

Puterman, M. L. 2005. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rigter, M.; Lacerda, B.; and Hawes, N. 2021. Risk-averse Bayes-adaptive reinforcement learning. In *Advances in Neural Information Processing Systems*.

Rockafellar, R. T.; Uryasev, S.; et al. 2000. Optimization of conditional value-at-risk. *Journal of risk*, 2: 21–42.

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

Santana, P.; Thiébaux, S.; and Williams, B. 2016. RAO*: An algorithm for chance-constrained POMDP's. *AAAI Conference on Artificial Intelligence*.

Shapiro, A.; Dentcheva, D.; and Ruszczyński, A. 2014. *Lectures on stochastic programming: modeling and theory*. SIAM.

Sobel, M. J. 1982. The variance of discounted Markov decision processes. *Journal of Applied Probability*, 19(4): 794–802.

Tamar, A.; Chow, Y.; Ghavamzadeh, M.; and Mannor, S. 2015. Policy gradient for coherent risk measures. In *Advances in Neural Information Processing Systems*, 1468–1476.

Tamar, A.; Chow, Y.; Ghavamzadeh, M.; and Mannor, S. 2016. Sequential decision making with coherent risk. *IEEE Transactions on Automatic Control*, 62(7): 3323–3338.

Tamar, A.; Glassner, Y.; and Mannor, S. 2015. Optimizing the CVaR via Sampling. In *AAAI Conference on Artificial Intelligence*, 2993–2999.

Tang, Y. C.; Zhang, J.; and Salakhutdinov, R. 2020. Worst cases policy gradients. *Conference on Robot Learning*.

Vamplew, P.; Dazeley, R.; Berry, A.; Issabekov, R.; and Dekker, E. 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning*, 84(1): 51–80.

Wang, S. S. 2000. A class of distortion operators for pricing financial and insurance risks. *Journal of risk and insurance*, 15–36.

Wray, K.; Zilberstein, S.; and Mouaddib, A.-I. 2015. Multi-objective MDPs with conditional lexicographic reward preferences. In *AAAI Conference on Artificial Intelligence*, volume 29.

3.1 Limitations and Future Work

There are several limitations to this work that prevent it from being more widely applicable. First, improving the expected value of the policy while maintaining the optimal CVaR is not possible for all problems. Our approach requires that for some reachable histories, it will be possible to take an alternate course of action that guarantees that the VaR of the original policy will not be exceeded. Whether this is possible depends on the structure of the specific problem. For example, we found that for some start-goal pairs of the autonomous navigation domain, our method resulted in no improvement. As seen in the paper, the improvement is only modest for the final version of the autonomous navigation domain that we used.

To guarantee that the CVaR remains optimal, our approach relies on exact solutions to the MDP that iterate over the entire state space. Therefore, it is unclear how our approach can be combined with approximate methods for solving MDPs such as sample-based search or reinforcement learning, where we will not be able to provide such a guarantee. Therefore, when considering more scalable approaches, it may be more sensible to focus on weighted combinations of CVaR and expected value [176], or risk measures that do not neglect the expected value [54].

This research hints at a direction for possible future work. We have shown that by switching between two policies that optimise for different objectives, we are able to derive an overall strategy that performs better for our chosen objective. An interesting direction for future work could be to investigate other approaches for composing together different policies with different objectives. For example, one could imagine optimising several policies for different levels of risk-aversion, and switching between the policies in a hierarchical manner. It will be a challenging question to determine in which situations, and for which optimisation objectives, better performance can be achieved by systematically switching between the policies with different levels of risk-aversion.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Planning for risk-aversion and expected value in MDPs.
Publication Status	□Published ☑Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Marc Rigter, Paul Duckworth, Bruno Lacerda and Nick Hawes (2022). Planning for risk-aversion and expected value in MDPs. International Conference on Automated Planning and Scheduling (ICAPS).

Student Confirmation

Student Name:	Marc Rigter		
Contribution to the Paper	I led the development of the idea in collaboration with Bruno Lacerda and Nick Hawes. I implemented the algorithm and all of the experiments except the road navigation domain. I led the writing in collaboration with Paul Duckworth, Bruno Lacerda, and Nick Hawes.		
Signature		Date	16th November 2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Nick Hawes		
Supervisor comments I confirm that Marc made a substantial contribution to the publication, and that the description above is accurate.			
Signature		Date	22nd November 2022

This completed form should be included in the thesis, at the end of the relevant chapter.

4

Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes

In the previous chapter, we assumed that the MDP representing the environment was known exactly. For the remaining chapters of this thesis, we no longer make such a strong assumption. As shown by Table 1.1, this, and all subsequent chapters, consider epistemic uncertainty. This means that we explicitly take into account a lack of knowledge about the environment.

The characteristics of the problem setting in this chapter are summarised in Table 1.1. In this chapter, we represent the uncertainty over the MDP in the form of a discrete set of possible MDPs. The true MDP is assumed to be one of the MDPs within this set. We consider the minimax regret objective, meaning that our goal is to find a *single* policy with the lowest maximum sub-optimality across the set of

Table 1.1: Summary of the attributes of each chapter.

	Chapter 3	**Chapter 4**	**Chapter 5**	**Chapter 6**	**Chapter 7**	**Chapter 8**
Prior knowledge	MDP fully known	MDP uncertainty set	Prior over MDPs	Fixed dataset	Fixed dataset	Expert demonstrator
Considers aleatoric uncertainty	Yes	No	Yes	No	Yes	No
Considers epistemic uncertainty	No	Yes	Yes	Yes	Yes	Yes
Risk or robustness	Risk	Robustness	Risk	Robustness	Risk	N/A
Paradigm	Tabular	Tabular	Tabular	Function Approx.	Function Approx.	Function Approx.

possible MDPs. Here, sub-optimality in each MDP refers to the expected value of the policy compared to the optimal expected value for each MDP. To achieve low maximum regret the policy must be near-optimal, in terms of expected value, in all of the plausible MDPs. Because the expected value is considered within each MDP, the approach presented in this chapter does not consider aleatoric uncertainty. Furthermore, because this work considers the *worst-case* sub-optimality in the uncertainty set, it achieves robustness to all scenarios in the uncertainty set.

We propose a dynamic programming approach for approximating the minimax regret. We first solve for the optimal expected value in each plausible MDP. Then, for each MDP we treat the difference between the optimal Q-value for a given action, and the optimal expected value, (i.e. the "advantage": $Q^*(s,a) - V^*(s)$) as the cost incurred for executing the action in that MDP. This cost represents the regret associated with each action in each MDP. Our approach performs minimax optimisation of this regret-based cost as an approximation for the minimax regret. To perform this optimisation, we use robust dynamic programming [12, 13] with the regret-based cost that we propose.[1]

Dynamic programming approaches to this type of minimax optimisation problem require that the MDP uncertainties are independent between each state-action pair (this type of uncertainty set is also called "rectangular"). However, for many problems there exist correlations between the uncertainties. For example, consider navigating the ocean in the presence of unknown currents. Knowing the direction of the current at one state informs what the current will be like at nearby states. However, these correlations cannot be modelled due to the independence assumption.

To help overcome this limitation we propose to define the policy using options, where each option is executed for n-steps. Within each option, dependencies between uncertainties are captured. In the ocean navigation problem, this means that within each option the direction of the uncertain ocean current is fixed. This requires

[1]In the original publication, we claimed that this approach was exact for independent/rectangular uncertainty sets. The authors later realised that this is incorrect, and this is indicated by the footnotes in the paper and in corrigendum included in the appendices for this paper. The authors have contacted AAAI to resolve this issue.

solving a mixed integer linear program to optimise minimax regret for each option. Between options, the uncertainties are assumed to be independent so that we can utilise dynamic programming and solve for each option separately. Depending on the value of n, this approach enables more realistic modelling of uncertainty at the expense of greater computation requirements.

We test our approach on problems where there are dependencies between the uncertainties, including a domain based on real ocean current forecasts. The results demonstrate that the method we propose outperforms existing approaches to the problem formulation addressed in this work. Additionally, the performance improves as the number of steps in each option is increased. This validates that our options-based approach enables more accurate modelling of the correlations between uncertainties.

Marc Rigter, Bruno Lacerda, and Nick Hawes (2021). Minimax regret optimisation for robust planning in uncertain Markov decision processes. *AAAI Conference on Artificial Intelligence* (AAAI).

Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes

Marc Rigter, Bruno Lacerda, Nick Hawes
Oxford Robotics Institute, University of Oxford, United Kingdom
{mrigter, bruno, nickh}@robots.ox.ac.uk

Abstract

The parameters for a Markov Decision Process (MDP) often cannot be specified exactly. Uncertain MDPs (UMDPs) capture this model ambiguity by defining sets which the parameters belong to. Minimax regret has been proposed as an objective for planning in UMDPs to find robust policies which are not overly conservative. In this work, we focus on planning for Stochastic Shortest Path (SSP) UMDPs with uncertain cost and transition functions. We introduce a Bellman equation to compute the regret for a policy. We propose a dynamic programming algorithm that utilises the regret Bellman equation, and show that it optimises minimax regret exactly for UMDPs with independent uncertainties. For coupled uncertainties, we extend our approach to use options to enable a trade off between computation and solution quality. We evaluate our approach on both synthetic and real-world domains, showing that it significantly outperforms existing baselines.

Introduction

Markov Decision Processes (MDPs) are a powerful tool for sequential decision making in stochastic domains. However, the parameters of an MDP are often estimated from limited data, and therefore cannot be specified exactly (Lacerda et al. 2019; Moldovan and Abbeel 2012). By disregarding model uncertainty and planning on the estimated MDP, performance can be much worse than anticipated (Mannor et al. 2004).

For example, consider an MDP model for medical decision making, where transitions correspond to the stochastic health outcomes for a patient as a result of different treatment options. An estimated MDP model for this problem can be generated from observed data (Schaefer et al. 2005). However, such a model does not capture the variation in transition probabilities due to patient heterogeneity: any particular patient may respond differently to treatments than the average due to unknown underlying factors. Additionally, such a model does not consider uncertainty in the model parameters due to limited data. As a result, Uncertain MDPs (UMDPs) have been proposed as more suitable model for domains such as medical decision making (Zhang, Steimle, and Denton 2017) where the model cannot be specified exactly.

UMDPs capture model ambiguity by defining an uncertainty set in which the true MDP cost and transition functions lie. In this work, we address offline planning for UMDPs. Most research in this setting has focused on optimising the expected value for the worst-case MDP parameters using robust dynamic programming (Iyengar 2005; Nilim and El Ghaoui 2005). However, this can result in overly conservative policies which perform poorly in the majority of possible scenarios (Delage and Mannor 2010). *Minimax regret* has been proposed as alternative metric for robust planning which is less conservative (Regan and Boutilier 2009; Xu and Mannor 2009). The aim is to find the policy with the minimum gap between its expected value and the optimal value over all possible instantiations of model uncertainty. However, optimising minimax regret is challenging and existing methods do not scale well.

In this work, we introduce a Bellman equation to decompose the computation of the regret for a policy into a dynamic programming recursion. We show that if uncertainties are *independent*, we can perform minimax value iteration using the regret Bellman equation to efficiently optimise minimax regret exactly.* To our knowledge, this is the first scalable exact algorithm for minimax regret planning in UMDPs with both uncertain cost and transition functions. To address problems with *dependent* uncertainties, we introduce the use of *options* (Sutton, Precup, and Singh 1999) to capture dependence over sequences of n steps. By varying n, the user may trade off computation time against solution quality.

Previous works have addressed regret-based planning in finite horizon problems (Ahmed et al. 2013, 2017), or problems where there is only uncertainty in the cost function (Regan and Boutilier 2009, 2010, 2011; Xu and Mannor 2009). We focus on the more general problem of Stochastic Shortest Path (SSP) UMDPs with uncertain cost and transition functions. The main contributions of this work are:

- Introducing a Bellman equation to compute the regret for a policy using dynamic programming.
- An efficient algorithm to optimise minimax regret exactly in models with independent uncertainties by performing minimax value iteration using our novel Bellman equation.*
- Proposing the use of options to capture dependencies between uncertainties to trade off solution quality against

*Following publication, the authors realised that this claim is incorrect. Please see the corrigendum.

computation for models with dependent uncertainties.

Experiments in both synthetic and real-world domains demonstrate that our approach considerably outperforms existing baselines.

Related Work

The worst-case expected value for a UMDP can be optimised efficiently with robust dynamic programming provided that the uncertainty set is convex, and the uncertainties are independent between states (Iyengar 2005; Nilim and El Ghaoui 2005; Wiesemann, Kuhn, and Rustem 2013). However, optimising for the worst-case expected value often results in overly conservative policies (Delage and Mannor 2010). This problem is exacerbated by the independence assumption which allows all parameters to be realised as their worst-case values simultaneously. Sample-based UMDPs represent model uncertainty with a finite set of possible MDPs, capturing dependencies between uncertainties (Adulyasak et al. 2015; Ahmed et al. 2013, 2017; Chen and Bowling 2012; Cubuktepe et al. 2020; Steimle, Kaufman, and Denton 2018). For sample-based UMDPs, dependent uncertainties can also be represented by augmenting the state space (Mannor, Mebel, and Xu 2016), however this greatly enlarges the state space even for a modest number of samples.

To compute less conservative policies, alternative planning objectives to worst-case expected value have been proposed. Possibilities include forgoing robustness and optimising average performance (Adulyasak et al. 2015; Steimle, Kaufman, and Denton 2018), performing chance-constrained optimisation under a known distribution of model parameters (Delage and Mannor 2010), and computing a Pareto-front for multiple objectives (Scheftelowitsch et al. 2017). *Minimax regret* has been proposed as an intuitive objective which is less conservative than optimising for the worst-case expected value (Xu and Mannor 2009), but can be considered robust as it optimises worst-case sub-optimality. Minimax regret in UMDPs where only the cost function is uncertain is addressed in (Regan and Boutilier 2009, 2010, 2011; Xu and Mannor 2009).

Limited research has addressed minimax regret in UMDP planning with *both* uncertain cost and transition functions. For sample-based UMDPs, the best stationary policy can be found by solving a Mixed Integer Linear Program (MILP), however this approach does not scale well (Ahmed et al. 2013). A policy-iteration algorithm is proposed by Ahmed et al. (2017) to find a policy with locally optimal minimax regret. However, this approach is only suitable for finite-horizon planning in which states are indexed by time step and the graph is acyclic. An approximation proposed by Ahmed et al. (2013) optimises minimax Cumulative Expected Myopic Regret (CEMR). CEMR myopically approximates regret by comparing local actions, rather than evaluating overall performance. Our experiments show that policies optimising CEMR often perform poorly for minimax regret. Unlike CEMR, our approach optimises minimax regret exactly for problems with independent uncertainties.

Regret is used to measure performance in reinforcement learning (RL) (eg. Jaksch, Ortner, and Auer 2010; Cohen et al. 2020; Tarbouriech et al. 2020). In the RL setting, the goal is to minimise the *total* regret, which is the total loss incurred throughout training over many episodes. In contrast, in our UMDP setting we plan offline to optimise the worst-case regret for a policy. This is the regret for a fixed policy evaluated over a single episode, assuming the MDP parameters are chosen adversarially. In RL, options (Sutton, Precup, and Singh 1999) have been utilised for learning robust policies with temporally extended actions (Mankowitz et al. 2018). In this work, we use options to capture dependencies between model uncertainties throughout the execution of each option.

Another approach to address MDPs which are not known exactly is Bayesian RL (Ghavamzadeh et al. 2015) which adapts the policy online throughout execution. In contrast to our setting, Bayesian RL typically does not address worst-case performance and requires access to a distribution over MDPs rather than a set. The offline minimax regret setting we consider is more appropriate for safety-critical domains such as medical decision making, where the policy must be scrutinised by regulators prior to deployment, and robustness to worst-case suboptimality is important.

Preliminaries

Definition 1. *An SSP MDP is defined as a tuple* $\mathcal{M} = (S, s_0, A, C, T, G)$. *$S$ is the set of states, $s_0 \in S$ is the initial state, A is the set of actions, $C : S \times A \times S \to \mathbb{R}$ is the cost function, and $T : S \times A \times S \to [0, 1]$ is the transition function. $G \subset S$ is the set of goal states. Each goal state $s_g \in G$ is absorbing and incurs zero cost.*

The expected cost of applying action a in state s is $\bar{C}(s, a) = \sum_{s' \in S} T(s, a, s') \cdot C(s, a, s')$. The minimum expected cost at state s is $\bar{C}^*(s) = \min_{a \in A} \bar{C}(s, a)$. A finite *path* is a finite sequence of states visited in the MDP. A *history-dependent* policy maps finite paths to a distribution over action choices. A *stationary* policy only considers the current state. A policy is *deterministic* if it chooses a single action at each step. The set of all policies is denoted Π. A policy is *proper at s* if it reaches $s_g \in G$ from s with probability 1. A policy is *proper* if it is proper at all states. In an SSP MDP, the following assumptions are made (Kolobov et al. 2012): a) there exists a proper policy, and b) every improper policy incurs infinite cost at all states where it is improper.

In this work, we aim to minimise the regret for a fixed policy over a single episode which is defined as follows.

Definition 2. *The regret for a policy $\pi \in \Pi$, denoted $reg(s_0, \pi)$, is defined as*

$$reg(s_0, \pi) = V(s_0, \pi) - V(s_0, \pi^*), \quad (1)$$

where $V(s, \pi)$ is the value of a policy π in state s according to the following Bellman equation,

$$V(s, \pi) = \sum_{a \in A} \pi(s, a) \cdot [\bar{C}(s, a) + \sum_{s' \in S} T(s, a, s') \cdot V(s', \pi)], \quad (2)$$

and π^* is the policy with minimal expected value. Intuitively, the regret for a policy is the expected suboptimality over a single episode. Ahmed et al. (2013, 2017) proposed Cumulative Expected Myopic Regret (CEMR) as a regret approximation.

Definition 3. *The CEMR of policy π at state s, denoted $cemr(s, \pi)$ is defined as*

$$cemr(s, \pi) = \sum_{a \in A} \pi(s, a) \cdot [\bar{C}(s, a) - \bar{C}^*(s) + \sum_{s' \in S} T(s, a, s') \cdot cemr(s', \pi)]. \quad (3)$$

$\bar{C}(s,a) - \bar{C}^*(s)$ is the gap between the expected cost of a, and the best expected cost for any action at s. CEMR is myopic, accumulating the local regret relative to the actions available at each state.

Uncertain MDPs We use the commonly employed sample-based UMDP definition (Adulyasak et al. 2015; Ahmed et al. 2013, 2017; Chen and Bowling 2012; Steimle, Kaufman, and Denton 2018). This representation captures dependencies between uncertainties because each sample represents an entire MDP. As we are interested in worst-case regret, we only require samples which provide adequate coverage over possible MDPs, rather than a distribution over MDPs.

Definition 4. *An SSP UMDP is defined by the tuple $(S, s_0, A, \mathcal{C}, \mathcal{T}, G)$. S, s_0, A, and G are defined as for SSP MDPs. $\mathcal{T} = \{T_1, T_2, \ldots, T_{|\xi|}\}$ denotes a finite set of possible transition functions and $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\xi|}\}$ denotes the associated set of possible cost functions. A sample of model uncertainty, ξ_q, is defined as $\xi_q = (C_q, T_q)$ where $C_q \in \mathcal{C}$, $T_q \in \mathcal{T}$. The set of samples is denoted ξ.*

We provide a definition for independent uncertainty sets, equivalent to the state-action rectangularity property introduced in (Iyengar 2005). Intuitively this means that uncertainties are decoupled between subsequent action choices.

Definition 5. *Set $\mathcal{T}$ is independent over state-action pairs if $\mathcal{T} = \times_{(s,a)\in S\times A}\mathcal{T}^{s,a}$ where $\mathcal{T}^{s,a}$ is the set of possible distributions over S after applying a in s, and $\times$ denotes the Cartesian product.*

The definition of independence for cost functions is analogous. In this work we wish to find π_{reg}, the policy which minimises the maximum regret over the uncertainty set.

Problem 1. *Find the minimax regret policy defined as*

$$\pi_{reg} = \underset{\pi\in\Pi}{\operatorname{argmin}} \max_{\xi_q\in\xi} reg_q(s_0, \pi), \tag{4}$$

where $reg_q(s_0, \pi)$ is the regret of π in the MDP corresponding to sample ξ_q. In general, stochastic policies are required to hedge against alternate possibilities (Xu and Mannor 2009), and history-dependent policies are required if uncertainties are dependent (Steimle, Kaufman, and Denton 2018; Wiesemann, Kuhn, and Rustem 2013). If only stationary deterministic policies are considered, a minimax regret policy can be computed exactly by solving a MILP (Ahmed et al. 2013). An approximation for minimax regret is to find the policy with minimax CEMR (Ahmed et al. 2013, 2017):

$$\pi_{cemr} = \underset{\pi\in\Pi}{\operatorname{argmin}} \max_{\xi_q\in\xi} cemr_q(s_0, \pi), \tag{5}$$

where $cemr_q(s_0, \pi)$ is the CEMR of π corresponding to ξ_q.

Our work is closely connected to the UMDP solution which finds the best expected value for the worst-case parameters (Iyengar 2005; Nilim and El Ghaoui 2005; Wiesemann, Kuhn, and Rustem 2013). We refer to the resulting policy as the *robust* policy. Assuming independent uncertainties per Def. 5, finding the robust policy can be posed as a Stochastic Game (SG) between the agent, and an adversary $\sigma^1 : S \times A \times \xi \to \{0, 1\}$ which responds to the action of the agent by applying the worst-case parameters at each step:

$$\pi_{robust} = \underset{\pi\in\Pi}{\operatorname{argmin}} \max_{\sigma^1} V(s_0, \pi). \tag{6}$$

The meaning of the superscript for σ^1 will become clear later.

For this problem, the optimal value function may be found via minimax Value Iteration (VI) and corresponds to a deterministic stationary policy for both players (Wiesemann, Kuhn, and Rustem 2013). For SSPs, convergence is guaranteed if: a) there exists a policy for the agent which is proper for all possible policies of the adversary, and b) for any states where π and σ are improper, the expected cost for the agent is infinite (Patek and Bertsekas 1999).

Regret Bellman Equation

Our first contribution is Proposition 1 which introduces a Bellman equation to compute the regret for a policy via dynamic programming. Full proofs of all propositions are in the full version of the paper (Rigter, Lacerda, and Hawes 2020).

Proposition 1. *(Regret Bellman Equation) The regret for a proper policy, π, can be computed via the following recursion*

$$reg(s,\pi) = \textstyle\sum_{a\in A}\pi(s,a)\cdot \\ \quad \big[Q^{gap}(s,a) + \sum_{s'\in S} T(s,a,s')\cdot reg(s',\pi)\big], \quad \textit{where} \tag{7}$$

$$Q^{gap}(s,a) = \big[\bar{C}(s,a) + \textstyle\sum_{s'\in S} T(s,a,s')\cdot V(s',\pi^*)\big] - V(s,\pi^*), \tag{8}$$

and $reg(s,\pi) = 0,\ \forall s \in G$.

Q^{gap} represents the suboptimality attributed to an s, a pair.

Proof sketch: unrolling Eq. 7-8 from s_0 for h steps we have

$$reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_a \pi(s_0,a)\Big[\bar{C}(s_0,a) + \\ \sum_{s'} T(s_0,a,s')\Big[\sum_a \pi(s',a)\Big[\bar{C}(s',a) + \ldots \Big[\sum_a \pi(s^{h\text{-}1},a)\Big[\bar{C}(s^{h\text{-}1},a) \\ + \sum_{s^h} T(s^{h\text{-}1},a,s^h)V(s^h,\pi^*) + \sum_{s^h} T(s^{h\text{-}1},a,s^h)reg(s^h,\pi)\Big]\Big]\ldots\Big]\Big]\Big]$$

Taking $h \to \infty$ we have $s^h \in G$ under the definition of a proper policy. Thus, $V(s^h,\pi^*) = reg(s^h,\pi) = 0$. Simplifying, we get $reg(s_0,\pi) = V(s_0,\pi) - V(s_0,\pi^*)$ which is the original definition for the regret of a policy. □

Minimax Regret Optimisation

In this section, we describe how Proposition 1 can be used to optimise minimax regret in UMDPs. We separately address UMDPs with independent and dependent uncertainties.

Exact Solution for Independent Uncertainties

To address minimax regret optimisation in UMDPs with independent uncertainties per Def. 5, we start by considering the following SG in Problem 2. At each step, the agent chooses an action, and the adversary $\sigma^1 : S \times A \times \xi \to \{0, 1\}$ reacts to this choice by choosing the MDP sample to be applied for that step to maximise the regret of the policy.

Problem 2. *Find the minimax regret policy in the stochastic game defined by*

$$\pi^1_{reg} = \underset{\pi\in\Pi}{\operatorname{argmin}} \max_{\sigma^1} reg(s_0, \pi). \tag{9}$$

Proposition 2. *If uncertainties are independent per Def. 5 then Problem 1 is equivalent to Problem 2.*

Proof sketch: For independent uncertainty sets, an adversary which chooses one set of parameters to be applied for the entire game is equivalent to an adversary which may change the parameters each step according to a stationary policy. □

Intuitively, this is because for any independent uncertainty set, fixing the parameters applied at one state-action pair does not restrict the set of parameters choices available at other state-action pairs. For problems of this form, deterministic stationary policies suffice (Iyengar 2005).

Problem 2 can be solved by applying minimax VI to the regret Bellman equation in Proposition 1. In the next section, we present Alg. 1 which solves a generalisation of Problem 2. The generalisation optimises minimax regret against an adversary, σ^n, that may change the parameters every n steps. To solve Problem 2, we apply Alg. 1 with $n = 1$. Proposition 2 shows that this optimises minimax regret exactly for UMDPs with independent uncertainty sets.*

Approx. Solutions for Dependent Uncertainties

For UMDPs with dependent uncertainties, optimising minimax regret exactly is intractable (Ahmed et al. 2017). A possible approach is to over-approximate the uncertainty by assuming independent uncertainties, and solve Problem 2. However, this gives too much power to the adversary, allowing parameters from different samples to be realised within the same game. Thus, the minimax regret computed under this assumption is an over-approximation, and the resulting policy may be overly conservative. In this section, we propose a generalisation of Problem 2 as a way to alleviate this issue. We start by bounding the maximum possible error of the over-approximation associated with solving Problem 2 for UMDPs with dependent uncertainties.

Proposition 3. *If the expected number of steps for π to reach $s_g \in G$ is at most H for any adversary:*

$$0 \leq \max_{\sigma^1} reg(s_0, \pi) - \max_{\xi_q \in \xi} reg_q(s_0, \pi) \leq (\delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H)H, \quad \text{where} \quad (10)$$

$$\begin{aligned} |\bar{C}_i(s,a) - \bar{C}_j(s,a)| &\leq \delta_C && \forall s \in S, a \in A, \xi_i \in \xi, \xi_j \in \xi \\ \textstyle\sum_{s'} |T_i(s,a,s') - T_j(s,a,s')| &\leq 2\delta_T && \forall s \in S, a \in A, \xi_i \in \xi, \xi_j \in \xi \\ |V_i(s,\pi^*) - V_j(s,\pi^*)| &\leq \delta_{V^*} && \forall s \in S, \xi_i \in \xi, \xi_j \in \xi \\ \bar{C}_i(s,a) &\leq C_{max} && \forall s \in S, a \in A, \xi_i \in \xi \end{aligned}$$

n-Step Options Prop. 3 shows that decoupling the uncertainties at every step over-approximates the maximum regret. We now introduce an approximation which is more accurate, but requires increased computation. Our approach is to approximate dependent uncertainties by decoupling the uncertainty at only every n steps. This results in Problem 3, a generalisation of Problem 2 where the agent chooses a policy to execute for n steps, and the adversary, σ^n, reacts by choosing the MDP sample to be applied for that n steps to maximise the regret. After executing n steps, the game transitions to a new state and the process repeats. Increasing n weakens the adversary by capturing dependence over each n step sequence. As $n \to \infty$ we recover the original minimax regret definition (Problem 1).

*Following publication, the authors realised that this claim is incorrect. Please see the corrigendum.

Problem 3. *Find the minimax regret policy in the stochastic game defined by*

$$\pi^n_{reg} = \operatorname*{argmin}_{\pi \in \Pi} \max_{\sigma^n} reg(s_0, \pi). \quad (11)$$

In the remainder of this section, we present our approach to solving Problem 3. We start by defining n-step options, an adaption of options (Sutton, Precup, and Singh 1999).

Definition 6. *An n-step option is a tuple $o = (\bar{s}, \pi^o, G^o, n)$, where $\bar{s}$ is the initiation state where the option may be selected, π^o is a policy, G^o is a set of goal states, and n is the number of steps.*

If an n-step option is executed at $\bar{s}$, the policy π^o is executed until one of two conditions is reached: either n steps pass, or a goal state $s_g \in G^o$ is reached. Hereafter, we assume that the goal states for all n-step options coincide with the goal states for the UMDP, $G^o = G$. The probability of reaching s' after executing option o in $\bar{s}$ is denoted by $\Pr(s'|\bar{s}, o)$.

We are now ready to define the n-step option MDP (n-MDP). The n-MDP is equivalent to the original MDP, except that we reason over options which represent policies executed for n steps in the original MDP. Additionally, the cost for applying option o at s in the n-MDP is equal to the regret attributed to applying π^o at s in the original MDP for n steps according to the regret Bellman equation in Proposition 1.

Definition 7. *An n-step option MDP (n-MDP) $\mathcal{M}^n$, of original SSP MDP $\mathcal{M}$, is defined by the tuple (S, s_0, O, C^o, T^o, G). S, s_0, and G are the same as in the original MDP. O is the set of possible n-step options. $T^o : S \times O \times S \to [0, 1]$ is the transition function for applying options, where $T^o(s, o, s') = \Pr(s'|s, o)$. The cost function, $C^o : S \times O \to \mathbb{R}$ is defined as:*

$$C^o(s, o) = V^n(s, \pi^o) + \textstyle\sum_{s' \in S} T^o(s, o, s') \cdot V(s', \pi^*) - V(s, \pi^*), \quad (12)$$

where $V^n(s, \pi^o)$ is the expected value for applying π^o for n steps starting in s.

The policy that selects options for the n-MDP is denoted π^n. We can convert a UMDP to a corresponding n-UMDP by converting each MDP sample in the UMDP to an n-MDP.

Proposition 4. *Problem 3 is equivalent to finding the robust policy (Eq. 6) for the n-UMDP.*

Proof sketch: The regret Bellman equation in Proposition 1 can equivalently be written for an n-MDP as

$$reg(s, \pi^n) = \sum_{o \in O} \pi^n(s, o)[C^o(s, o) + \sum_{s' \in S} T^o(s, o, s') reg(s', \pi^n)]. \quad (13)$$

This is an MDP Bellman equation using the cost and transition functions for the n-MDP. Therefore, finding the minimax regret according to Problem 3 is equivalent to finding the minimax expected cost for the n-UMDP. This is optimised by the robust policy for the n-UMDP. □

Minimax Value Iteration Prop. 4 means we can use minimax VI on the n-MDP to solve Problem 3 via the recursion

$$reg(s, \pi^n) = \min_{o \in O} \max_{\xi_q \in \xi} [C^o_q(s, o) + \textstyle\sum_{s' \in S} T^o_q(s, o, s') reg(s', \pi^n)]. \quad (14)$$

Algorithm 1 Minimax Value Iteration for Minimax Regret Optimisation with n-Step Options

1: compute $V_q(s, \pi_q^*)\ \forall s \in S, \xi_q \in \xi$
2: $reg(s, \pi^n) \leftarrow 0,\ \forall s \in S$
3: **repeat**
4: $\quad \Delta \leftarrow 0$
5: $\quad$ **for** $\bar{s} \in S$ **do**
6: $\quad\quad reg_{old} \leftarrow reg(\bar{s}, \pi^n)$
7: $\quad\quad reg(\bar{s}, \pi^n) \leftarrow \min_{o \in O} \max_{\xi_q \in \xi} [C_q^o(\bar{s}, o) + \sum_{s'} T_q^o(\bar{s}, o, s') \cdot reg(s', \pi^n)]$ (Eq. 14)
8: $\quad\quad \pi^n(\bar{s}) \leftarrow \text{argmin}_{o \in O} \max_{\xi_q \in \xi} [C_q^o(\bar{s}, o) + \sum_{s'} T_q^o(\bar{s}, o, s') \cdot reg(s', \pi^n)]$ (Eq. 14)
9: $\quad\quad \Delta \leftarrow \max(\Delta, |reg(\bar{s}, \pi^n) - reg_{old}|)$
10: $\quad$ **end for**
11: **until** $\Delta < \epsilon$

The results for minimax VI apply (Iyengar 2005; Nilim and El Ghaoui 2005), and therefore the optimal policy is stationary and deterministically chooses options, $\pi^n : S \times O \rightarrow \{0, 1\}$. To guarantee convergence, we apply a perturbation by adding a small scalar $\kappa > 0$ to the cost in Eq. 12. It can be shown that in the limit as $\kappa \rightarrow 0^+$, the resulting value function approaches the exact solution (Bertsekas 2018).

Algorithm 1 presents pseudocode for the minimax VI algorithm. In Line 1, we start by computing the optimal value in each of the MDP samples using standard VI. This is necessary to compute the contribution to the regret of any action according to Proposition 1. Line 2 initialises the values for the minimax regret of the policy to zero at all states. In Lines 5-10 we sweep through each state until convergence. At each state we update both the minimax regret value and the option chosen by the policy, according to the Bellman backup defined by Eq. 14. Solving Equation 14 is non-trivial, and we formulate a means to solve it in the following subsection.

Eq. 15 of Prop. 5 establishes that for dependent uncertainties, $\max_{\sigma^n} reg(s_0, \pi)$ is an upper bound on the maximum regret for the policy for any n. Eq. 16 shows that if we increase n by a factor k and optimise the policy using Algorithm 1, we are guaranteed to equal or decrease this upper bound. Our experiments demonstrate that in practice increasing n improves performance substantially.

Proposition 5. *For dependent uncertainty sets,*

$$\max_{\sigma^n} reg(s_0, \pi) - \max_{\xi_q \in \xi} reg_q(s_0, \pi) \geq 0 \quad \forall\, n \in \mathbb{N}, \quad (15)$$

$$\min_{\pi^n} \max_{\sigma^n} reg(s_0, \pi^n) \geq \min_{\pi^{kn}} \max_{\sigma^{kn}} reg(s_0, \pi^{kn}) \quad \forall\, n, k \in \mathbb{N}. \quad (16)$$

Optimising the Option Policies To perform minimax VI in Algorithm 1, we repeatedly solve the Bellman equation defined by Eq. 14. Eq. 14 corresponds to finding an option policy by solving a finite-horizon minimax regret problem with dependent uncertainties. Because of the dependence over the n steps, the optimal option policy may be history-dependent (Steimle, Kaufman, and Denton 2018; Wiesemann, Kuhn, and Rustem 2013). Intuitively, this is because for dependent uncertainty sets, the history may provide information about the uncertainty set at future stages. To maintain scalability, whilst still incorporating some memory into the option policy, we opt to consider option policies which depend only on the state and time step, t. Therefore, the optimisation problem in Eq. 14 can be written as Table 1.

min $reg(\bar{s}, \pi^n)$ **s. t.**

$$reg(\bar{s}, \pi^n) \geq V_q^n(\bar{s}, t) - V_q(\bar{s}, \pi_q^*) + c_q(\bar{s}, t), \ \ \forall \xi_q, t = 0 \quad (17)$$

$$c_q(s, a, t) = \textstyle\sum_{s'} T_q(s, a, s') \cdot [reg(s', \pi^n) + V_q(s', \pi_q^*)], \quad \forall s \in S_{\bar{s},t}^q, a, \xi_q, t = n - 1 \quad (18)$$

$$c_q(s, a, t) = \textstyle\sum_{s'} T_q(s, a, s') \cdot c_q(s', t + 1), \quad \forall s \in S_{\bar{s},t}^q, a, \xi_q, t < n - 1 \quad (19)$$

$$c_q(s, t) = \textstyle\sum_a \pi^o(s, a) \cdot c_q(s, a, t), \quad \forall s \in S_{\bar{s},t}^q, \xi_q, t \leq n - 1 \quad (20)$$

$$V_q^n(s, a, t) = \bar{C}_q(s, a), \qquad \forall s \in S_{\bar{s},t}^q, a, \xi_q, t = n - 1 \quad (21)$$

$$V_q^n(s, a, t) = \bar{C}_q(s, a) + \textstyle\sum_{s'} T_q(s, a, s') \cdot V_q^n(s', t + 1), \quad \forall s \in S_{\bar{s},t}^q, a, \xi_q, t < n - 1 \quad (22)$$

$$V_q^n(s, t) = \textstyle\sum_a \pi^o(s, a) \cdot V_q^n(s, a, t), \quad \forall s \in S_{\bar{s},t}^q, \xi_q, t \leq n - 1 \quad (23)$$

Table 1: Formulation of the optimisation problem over option policies in Equation 14.

In Table 1, we optimise $reg(\bar{s}, \pi^n)$, the updated value for the minimax regret at $\bar{s}$. The other optimisation variables are those denoted by c_q, V_q^n, and π^o. The optimal value in each sample, $V_q(s, \pi_q^*)$, is precomputed in Line 1 of Alg 1. $reg(s', \pi^n)$ is the current estimate of the minimax regret for π^n at each state, and is initialised to zero in Line 2 of Alg 1.

The constraints in Table 1 represent the following. The set $S_{\bar{s},t}^q$ contains all the states reachable in exactly t steps, starting from $\bar{s}$, in sample ξ_q. Eq. 17 corresponds to the regret Bellman equation for options in Eq. 12-13, where the inequality over all ξ_q enforces minimising the maximum regret. The variables denoted $c_q(s, t)$ represent the expected cumulative part of the minimax regret in Eq. 12-13 resulting from the expected state distribution after applying π^o in sample ξ_q for the horizon of n steps. Constraint equations 18-20 propagate these c_q values over the n step horizon. The variables denoted $V_q^n(s, t)$ represent the expected value over the n-step horizon of π^o at time t in sample ξ_q, and the computation of the expected value is enforced by the constraints in Eq. 21-23.

In the supplementary material, we provide linearisations for the nonlinear constraints in Eq. 20 and 23. We consider both deterministic and stochastic option policies, as due to the dependent uncertainties the optimal option policy may be stochastic (Wiesemann, Kuhn, and Rustem 2013). The solution is exact for deterministic policies, and for stochastic policies a piecewise linear approximation is required.

Discussion

Complexity For SSP MDPs with positive costs, the number of iterations required for VI to converge within residual ϵ is bounded by $\mathcal{O}(||V^*||^2|S|^2/g^2 + (\log ||V^*|| + \log \epsilon)||V^*||\,|S|/g)$, where g is the minimum cost, V^* is the optimal value, and $||x||$ is the L_∞ norm of x (Bonet 2007). In our problem, the minimum cost is κ. During each VI sweep, we solve Eq. 14 $|S|$ times by optimising Table 1. Therefore, Table 1 must be solved $\mathcal{O}(||reg^*||^2|S|^3/\kappa^2 + (\log ||reg^*|| + \log \epsilon)||reg^*||\,|S|^2/\kappa)$ times, where reg^* is the optimal minimax regret for Problem 3. To assess the complexity of optimising the model in Table 1, assume the MDP branching factor is k. Then the number reachable states in n steps is

$\mathcal{O}(k^n)$, and the size of the model is $\mathcal{O}(|A||\xi|nk^n)$. MILP solution time is exponential, and therefore complexity is $\mathcal{O}(exp(|A||\xi|nk^n))$. Crucially, $|S|$ is not in the exponential.

Sampling This approach requires a finite set of samples, yet for some problems there are infinite possible MDP instantiations. To optimise worst-case regret, we need samples which provide adequate coverage over possible MDP instantiations so that the resulting worst-case solution generalises to all possible MDPs. Where necessary, we use the sampling approach proposed for this purpose in (Ahmed et al. 2013).

Action Pruning To reduce the size of the problem in Table 1, we can prune out actions that are unlikely to be used by the minimax regret policy. We propose the following pruning method, analagous to the approach proposed by Lacerda, Parker, and Hawes (2017). The policy with optimal expected cost, π^*_q, is computed for each $\xi_q \in \xi$ to create the set of optimal policies $\Pi^*_\xi = \{\pi^*_1, \pi^*_2, \ldots, \pi^*_{|\xi|}\}$. We build the pruned UMDP by removing transitions $T_q(s, a, s')$ for all ξ_q, for which a is not in $\pi(s)$ for some policy $\pi \in \Pi^*_\xi$. The intuition is that actions which are directed towards the goal are likely to be included in at least one of the optimal policies. Therefore, this pruning process removes actions which are not directed towards the goal, and are unlikely to be included in a policy with low maximum regret.

Evaluation

We evaluate the following approaches on three domains with dependent uncertainties:

- *reg*: the approach presented in this paper.
- *cemr*: the state of the art approach from (Ahmed et al. 2013, 2017) which we have extended for n-step options.
- *robust*: the standard robust dynamic programming solution in Eq. 6 (Iyengar 2005; Nilim and El Ghaoui 2005).
- *MILP*: the optimal stationary deterministic minimax regret policy computed using the MILP in (Ahmed et al. 2013). We do not compare against the stochastic version as we found it did not scale beyond very small problems.
- *Averaged MDP*: this benchmark averages the cost and transition parameters across the MDP samples and computes the optimal policy in the resulting MDP (Adulyasak et al. 2015; Chen and Bowling 2012; Delage and Mannor 2010).
- *Best MDP Policy*: computes the optimal policy set, Π^*_ξ. For each policy, $\pi \in \Pi^*_\xi$, the maximum regret is evaluated for all $\xi_q \in \xi$. The policy with lowest maximum regret is selected.

We found that action pruning reduced computation time for *reg*, *cemr*, and *MILP* without significantly harming performance. Therefore we only present results for these approaches with pruning. We write (s) for stochastic (d) for deterministic option policies. MILPs are solved using Gurobi, and all other processing is performed in Python. Computation times are reported for a 3.2 GHz Intel i7 processor. For further details on the experimental domains see Rigter, Lacerda, and Hawes (2020).

Medical Decision Making Domain We test on the medical domain from Sharma et al. (2019). The state, $(h, d) \in S$ comprises of two factors: the health of the patient, $h \in \{0, \ldots, 19\}$, and the day $d \in \{0, \ldots, 6\}$. At any state, one of three actions can be applied, each representing different treatments. In each MDP sample the transition probabilities for each treatment differ, corresponding to different responses by patients with different underlying conditions. The health of the patient on the final day determines the cost received.

Disaster Rescue Domain We adapt this domain from the UMDP literature (Adulyasak et al. 2015; Ahmed et al. 2013, 2017) to SSP MDPs. An agent navigates an 8-connected grid which contains swamps and obstacles by choosing from 8 actions. Nominally, for each action the agent transitions to the corresponding target square with $p = 0.8$, and to the two adjacent squares with $p = 0.1$ each. If the target, or adjacent squares are obstacles, the agent transitions to that square with probability 0.05. If the square is a swamp, the cost is sampled uniformly in $[1, 2]$. The cost for entering any other state is 0.5. The agent does not know the exact locations of swamps and obstacles, and instead knows regions where they may be located. To construct a sample, a swamp and obstacle is sampled uniformly from each swamp and obstacle region respectively. Fig. 1 (left) illustrates swamp and obstacle regions for a particular UMDP. Fig. 1 (right) illustrates a possible sample corresponding to the same UMDP.

Underwater Glider Domain An underwater glider navigates to a goal location subject to uncertain ocean currents from real-world forecasts. For each UMDP, we sample a region of the Norwegian sea, and sample the start and goal location. The mission will be executed between 6am and 6pm, but the exact time is unknown. As such, the navigation policy must perform well during all ocean conditions throughout the day. We construct 12 MDP samples, corresponding to the ocean current forecast at each hourly interval. Each state is a grid cell with 500m side length. There are 12 actions corresponding to heading directions. The cost for entering each state is sampled in [0.8, 1]. An additional cost of 3 is added for entering a state where the water depth is < 260m and current is > 0.12m/s, penalising operation in shallow water with strong currents. The forecast used was for May 1st 2020 and is available online at https://marine.copernicus.eu/.

Experiments

Maximum Regret For the medical domain, each method was evaluated for 250 different randomly generated UMDPs. For the other two domains, each method was evaluated for a range of problem sizes, and each problem size was repeated for 25 different randomly generated UMDPs. For each disaster rescue and medical decision making UMDP, ξ consisted of 15 samples selected using the method from (Ahmed et al. 2013, 2017). In underwater glider, ξ consisted of the 12 samples corresponding to each hourly weather forecast. For each method, we include results where the average computation time was <600s. For the resulting policies, the maximum regret over all samples in ξ was computed. Each max regret value was normalised between [0, 1] by dividing by the worst max regret for that UMDP across each of the methods. The normalised values were then averaged over all 25/250 runs, and displayed in Fig 3 and the top row of Table 2.

Generalisation Evaluation For sample-based UMDPs, the policy is computed from a finite set of samples. To assess generalisation to any possible MDP instantiation, we

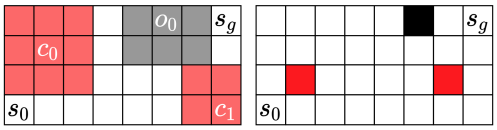


Figure 1: Illustration of an example UMDP in disaster rescue. Left: shaded regions indicate possible swamp and obstacle locations. Right: possible UMDP sample.

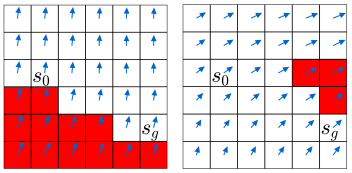


Figure 2: Illustration of two UMDP samples in glider domain. Arrows indicate ocean currents. Red squares indicate states with additional cost.

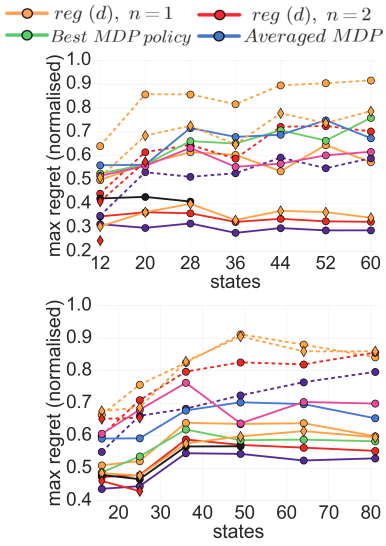


Figure 3: Mean normalised max regret. Top: disaster rescue, bottom: glider.

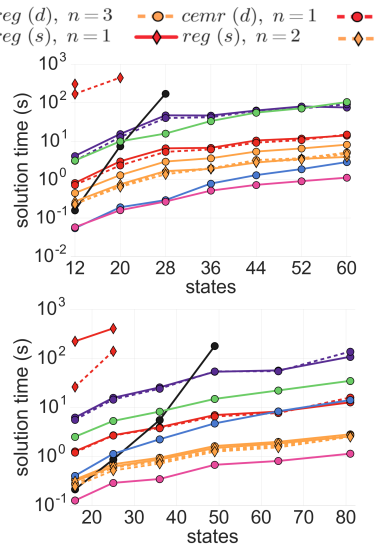


Figure 4: Mean solution times. Top: disaster rescue, bottom: glider.

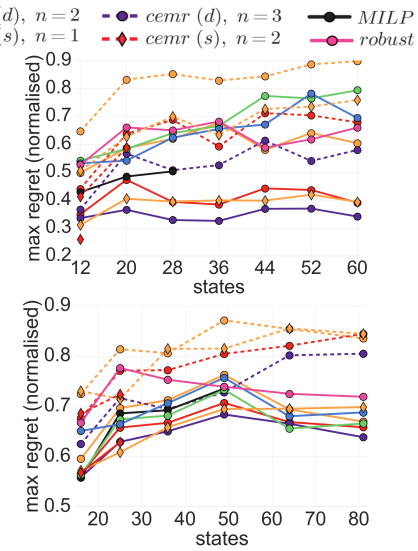


Figure 5: Mean normalised max regret on test set. Top: disaster rescue, bottom: glider.

Method											
max reg	0.596; 0.22	0.538; 0.18	**0.497**; 0.16	0.906; 0.11	0.885; 0.13	0.880; 0.12	0.557; 0.20	0.596; 0.22	0.641; 0.23	0.529; 0.15	0.846; 0.12
time (s)	5.03; 0.40	21.1; 2.3	77.6; 18	4.24; 0.33	20.0; 2.4	66.9; 16	103; 7.9	3.75; 0.08	**0.391**; 0.03	4.64; 0.35	3.96; 0.31
max reg test set	0.674; 0.19	0.636; 0.18	0.625; 0.18	0.870; 0.13	0.855; 0.14	0.852; 0.13	0.706; 0.20	0.677; 0.20	0.715; 0.19	**0.574**; 0.12	0.814;0.13

Table 2: Mean normalised maximum regret in the medical domain, averaged over 250 UMDPs. Methods which did not find a solution within an average time of 600s are not included in the table. Format: mean; standard deviation.

evaluated the maximum regret on a larger set of 100 random samples. For disaster rescue and medical decision making, the samples were generated using the procedure outlined. For underwater glider, we generated more samples by linearly interpolating between the 12 forecasts and adding Gaussian noise to the ocean current values ($\sigma = 2\%$ of the ocean current velocity). The average maximum regret for this experiment is shown in Fig. 5 and the bottom row of Table 2.

Results A table of p-values is included in the supplementary material which shows that most differences in performance between methods are statistically significant, with the exception of those lines in the figures which overlap. Fig. 3 shows that in general, our approach significantly outperforms the baselines with the exception of *MILP*, which also has good performance. However, *MILP* scales poorly as indicated by Fig. 4, and failed to find solutions in the medical domain within the 600s time limit. *MILP* finds the optimal deterministic stationary policy considering full dependence between uncertainties. However, the performance of our approach improves significantly with increasing n, and outperforms *MILP* with larger n. This indicates that the limited memory of the non-stationary option policies is crucial for strong performance. For the same n, stochastic option policies improve performance in both domains. However, the poor scalability of stochastic policies indicates that deterministic options with larger n are preferable. Across all domains the current state of the art method, CEMR with $n = 1$, performed poorly. Performance of CEMR improved somewhat by extending it to use our options framework ($n > 1$). The poor performance of CEMR can be attributed to the fact that CEMR myopically approximates the maximum regret by calculating the performance loss compared to local actions, which may be a poor estimate of the suboptimality over the entire policy. In contrast, our approach optimises the maximum regret using the recursion given in Prop. 1 which computes the contribution of each action to the regret for the policy exactly by comparing against the optimal value function in each sample.

The generalisation results in Fig. 5 and Table 2 show strong performance of our approach for disaster rescue and the medical domain on the larger test set. In the glider domain, there is more overlap between methods however our approach with larger n still tends to perform the best. This verifies that in domains with a very large set of possible MDPs, a viable

approach is to use our method to find a policy with a smaller set of MDPs and this will generalise well to the larger set.

Conclusion

We have presented an approach for minimax regret optimisation in offline UMDP planning. Our algorithm solves this problem efficiently and exactly in problems with independent uncertainties.* To address dependent uncertainties we have proposed using options to capture dependence over sequences of n steps and tradeoff computation and solution quality. Our results demonstrate that our approach offers state-of-the-art performance. In future work, we wish to improve the scalability of our approach by using function approximation.

Acknowledgments

This work was supported by UK Research and Innovation and EPSRC through the Robotics and Artificial Intelligence for Nuclear (RAIN) research hub [EP/R026084/1], the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services.

References

Adulyasak, Y.; Varakantham, P.; Ahmed, A.; and Jaillet, P. 2015. Solving uncertain MDPs with objectives that are separable over instantiations of model uncertainty. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Ahmed, A.; Varakantham, P.; Adulyasak, Y.; and Jaillet, P. 2013. Regret based robust solutions for uncertain Markov decision processes. In *Advances in Neural Information Processing Systems*.

Ahmed, A.; Varakantham, P.; Lowalekar, M.; Adulyasak, Y.; and Jaillet, P. 2017. Sampling based approaches for minimizing regret in uncertain Markov decision processes. *Journal of Artificial Intelligence Research* 59.

Bagnell, J. A.; Ng, A. Y.; and Schneider, J. G. 2001. Solving uncertain Markov decision processes. Technical report, Carnegie Mellon University.

Bertsekas, D. P. 2018. *Abstract dynamic programming*. Athena Scientific.

Bonet, B. 2007. On the speed of convergence of value iteration on stochastic shortest-path problems. *Mathematics of Operations Research* 32(2).

Chen, K.; and Bowling, M. 2012. Tractable objectives for robust policy optimization. In *Advances in Neural Information Processing Systems*.

Cohen, A.; Kaplan, H.; Mansour, Y.; and Rosenberg, A. 2020. Near-optimal regret bounds for stochastic shortest path. *International Conference on Machine Learning* .

Cubuktepe, M.; Jansen, N.; Junges, S.; Katoen, J.-P.; and Topcu, U. 2020. Scenario-Based Verification of Uncertain MDPs. In *Tools and Algorithms for the Construction and Analysis of Systems*, 287–305. Springer International Publishing.

*Following publication, the authors realised that this claim is incorrect. Please see the corrigendum.

Delage, E.; and Mannor, S. 2010. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations research* 58(1).

Ghavamzadeh, M.; Mannor, S.; Pineau, J.; and Tamar, A. 2015. Bayesian Reinforcement Learning: A Survey. *Foundations and Trends in Machine Learning* 8(5–6): 359–483.

Ghavamzadeh, M.; Petrik, M.; and Chow, Y. 2016. Safe policy improvement by minimizing robust baseline regret. *Advances in Neural Information Processing Systems* 29.

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

Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* 11(Apr): 1563–1600.

Kolobov, A.; et al. 2012. *Planning with Markov decision processes: An AI perspective*. Morgan & Claypool Publishers.

Lacerda, B.; Faruq, F.; Parker, D.; and Hawes, N. 2019. Probabilistic planning with formal performance guarantees for mobile service robots. *The International Journal of Robotics Research* 38(9).

Lacerda, B.; Parker, D.; and Hawes, N. 2017. Multi-objective policy generation for mobile robots under probabilistic time-bounded guarantees. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*.

Liu, L.; and Sukhatme, G. S. 2018. A solution to time-varying Markov decision processes. *IEEE Robotics and Automation Letters* 3(3): 1631–1638.

Mankowitz, D. J.; Mann, T. A.; Bacon, P.-L.; Precup, D.; and Mannor, S. 2018. Learning robust options. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Mannor, S.; Mebel, O.; and Xu, H. 2016. Robust MDPs with k-rectangular uncertainty. *Mathematics of Operations Research* 41(4): 1484–1509.

Mannor, S.; Simester, D.; Sun, P.; and Tsitsiklis, J. N. 2004. Bias and variance in value function estimation. In *Proceedings of the Twenty-First International Conference on Machine Learning*.

Moldovan, T. M.; and Abbeel, P. 2012. Risk aversion in Markov decision processes via near optimal Chernoff bounds. In *Advances in Neural Information Processing Systems*.

Nilim, A.; and El Ghaoui, L. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* 53(5).

Patek, S. D.; and Bertsekas, D. P. 1999. Stochastic shortest path games. *SIAM Journal on Control and Optimization* 37(3).

Regan, K.; and Boutilier, C. 2009. Regret-Based Reward Elicitation for Markov Decision Processes. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*.

Regan, K.; and Boutilier, C. 2010. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Regan, K.; and Boutilier, C. 2011. Robust online optimization of reward-uncertain MDPs. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Rigter, M.; Lacerda, B.; and Hawes, N. 2020. Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes. *arXiv preprint* ArXiv: 2012.04626.

Schaefer, A. J.; Bailey, M. D.; Shechter, S. M.; and Roberts, M. S. 2005. Modeling medical treatment using Markov decision processes. In *Operations research and health care*, 593–612. Springer.

Scheftelowitsch, D.; Buchholz, P.; Hashemi, V.; and Hermanns, H. 2017. Multi-objective approaches to Markov decision processes with uncertain transition parameters. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*.

Sharma, A.; Harrison, J.; Tsao, M.; and Pavone, M. 2019. Robust and adaptive planning under model uncertainty. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Steimle, L. N.; Kaufman, D. L.; and Denton, B. T. 2018. Multi-model Markov decision processes. *Optimization Online* .

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112.

Tarbouriech, J.; Garcelon, E.; Valko, M.; Pirotta, M.; and Lazaric, A. 2020. No-Regret Exploration in Goal-Oriented Reinforcement Learning. *International Conference on Machine Learning* .

Wiesemann, W.; Kuhn, D.; and Rustem, B. 2013. Robust Markov decision processes. *Mathematics of Operations Research* 38(1): 153–183.

Williams, H. P. 2013. *Model building in mathematical programming*. John Wiley & Sons.

Xu, H.; and Mannor, S. 2009. Parametric regret in uncertain Markov decision processes. In *Proceedings of the 48h IEEE Conference on Decision and Control*. IEEE.

Zhang, Y.; Steimle, L.; and Denton, B. 2017. Robust Markov decision processes for medical treatment decisions. *Optimization online* .

4.1 Limitations and Future Work

A key issue with this piece of work is that some of the theoretical analysis presented in the original version of this paper was later found to be incorrect, as indicated in the footnotes, and in the corrigendum included in the appendices. Specifically, the approach we proposed does not give an exact solution for minimax regret for independent/rectangular uncertainty sets. As noted, we have contacted the publisher to amend this. Thankfully, this does not appear to be a major shortcoming as the independent uncertainty assumption is usually unrealistic, and problems with independent uncertainties are not the focus of this work. Furthermore, our experimental results show that the approach we proposed empirically outperforms existing approaches.

One of the limitations of our approach is that finding each option policy requires solving a mixed integer linear program. This prevents our method from scaling to longer option policies that capture greater dependencies between the uncertainties, as these sub-problems become too complex to solve.

An obvious next step for this work would be to try and apply the proposed ideas to larger problems by utilising methods based on reinforcement learning with function approximation. Rather than specifying an MDP uncertainty set, for more complex problems it would be more natural to consider some unknown parameter(s) that govern the dynamics. For example, in robotics an example of an unknown parameter could be the friction coefficient for the contact between the robot and the ground.

To adapt our approach for approximating minimax regret, we could first find the optimal Q-function for a number of samples of the unknown parameter using reinforcement learning. Then, we could optimise a final policy to minimise the regret-based cost that we proposed in this work. The regret-based cost would be computed by evaluating the maximum sub-optimality of each action across the optimal Q-functions for each of the plausible parameter values. Optimising a policy based on our proposed regret-based cost would approximate the minimax regret criterion. The first step of finding the optimal Q-function for a range of parameter values

would be computationally demanding for large problems. However, this could be alleviated by including the parameter value in the state space, and learning a single Q-function, thereby enabling generalisation between different parameter values.

It would be interesting to investigate whether this proposed approach outperforms RL approaches that optimise for the best expected value in the worst-case scenario [98, 77, 78], that might suffer from being overly conservative.

This proposed approach would implicitly assume that the uncertainties are independent between each step, allowing for the value of the unknown parameter to change each step. To alleviate this assumption, it would be interesting to investigate whether the idea that we proposed to utilise options with fixed uncertainty can be adapted to the deep RL setting.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Minimax regret optimisation for robust planning in uncertain Markov decision processes.
Publication Status	□Published ☑Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Marc Rigter, Bruno Lacerda, and Nick Hawes (2021). Minimax regret optimisation for robust planning in uncertain Markov decision processes. AAAI Conference on Artificial Intelligence (AAAI).

Student Confirmation

Student Name:	Marc Rigter		
Contribution to the Paper	I developed the idea in collaboration with Bruno Lacerda and Nick Hawes. I implemented the algorithm and ran the experiments. I led the writing in collaboration with Bruno Lacerda and Nick Hawes.		
Signature		Date	16th November 2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Nick Hawes		
Supervisor comments	I confirm that Marc made a substantial contribution to the publication, and that the description above is accurate.		
Signature		Date	22nd November 2022

This completed form should be included in the thesis, at the end of the relevant chapter.

5

Risk-Averse Bayes-Adaptive Reinforcement Learning

In the previous chapter, we assumed that the prior knowledge about the environment was specified by a set of candidate MDPs, and that this knowledge did not improve after interacting with the environment during execution. In this chapter, we now assume that the knowledge about the environment can be improved during the course of each episode. Additionally, the previous chapter disregarded aleatoric uncertainty, as it considered the expected value in each plausible MDP. In this chapter, we now propose to address both aleatoric and epistemic uncertainty under a single framework by optimising a risk-averse objective. These characteristics are summarised in Table 1.1.

Our approach builds upon the Bayesian formulation of model-based reinforcement learning, where we model a belief distribution over the unknown MDP. We consider the Bayes-Adaptive MDP (BAMDP) which augments the state space of the original

Table 1.1: Summary of the attributes of each chapter.

	Chapter 3	**Chapter 4**	**Chapter 5**	**Chapter 6**	**Chapter 7**	**Chapter 8**
Prior knowledge	MDP fully known	MDP uncertainty set	Prior over MDPs	Fixed dataset	Fixed dataset	Expert demonstrator
Considers aleatoric uncertainty	Yes	No	Yes	No	Yes	No
Considers epistemic uncertainty	No	Yes	Yes	Yes	Yes	Yes
Risk or robustness	Risk	Robustness	Risk	Robustness	Risk	N/A
Paradigm	Tabular	Tabular	Tabular	Function Approx.	Function Approx.	Function Approx.

MDP with the belief over the MDP. In this work, we propose to optimise a risk-sensitive objective, specifically CVaR in the BAMDP.

Our motivation for this approach is that it mitigates both epistemic and aleatoric uncertainty under a single framework, as indicated in Table 1.1. The belief distribution over MDPs captures the epistemic uncertainty about the MDP dynamics. The dynamics of the BAMDP are influenced by both the belief distribution and the aleatoric uncertainty within each plausible MDP. Therefore, optimising for risk in the BAMDP must avoid poor outcomes that are possible due to either source of uncertainty. To make this point more concretely, we show that CVaR in a BAMDP is equivalent to the expected value under an adversarial perturbation to both the prior distribution over MDPs, and the transition probabilities within each plausible MDP.

We extend an existing approach to CVaR optimisation [56]. We model the problem as a turn-based two-player zero-sum game, which is played over the fully-observable BAMDP. In the game, the adversary modifies the transition probabilities in the BAMDP to reduce the expected return of the agent. This makes it more likely that the agent experiences transitions in the BAMDP that either a) receive a low rewards, or b) result in the agent transitioning to a belief state where the worst possible MDPs are more likely. This results in risk-averse behaviour from the agent, as the agent acts optimally under the assumption that bad transitions, or bad MDPs are more likely than they are under the prior belief.

We propose an algorithm based on two-player Monte Carlo tree search to approximately solve this game. Due to the continuous action space of the adversary, this problem is computationally challenging. To improve scalability we use progressive widening with Bayesian optimisation [177] to gradually expand the action space of the adversary. We demonstrate that this approach outperforms existing baselines on two small domains.

Marc Rigter, Bruno Lacerda and Nick Hawes (2021). Risk-averse Bayes-adaptive reinforcement learning. *Advances in Neural Information Processing Systems* (NeurIPS).

Risk-Averse Bayes-Adaptive Reinforcement Learning

Marc Rigter
Oxford Robotics Institute
University of Oxford
mrigter@robots.ox.ac.uk

Bruno Lacerda
Oxford Robotics Institute
University of Oxford
bruno@robots.ox.ac.uk

Nick Hawes
Oxford Robotics Institute
University of Oxford
nickh@robots.ox.ac.uk

Abstract

In this work, we address risk-averse Bayes-adaptive reinforcement learning. We pose the problem of optimising the conditional value at risk (CVaR) of the total return in Bayes-adaptive Markov decision processes (MDPs). We show that a policy optimising CVaR in this setting is risk-averse to both the epistemic uncertainty due to the prior distribution over MDPs, and the aleatoric uncertainty due to the inherent stochasticity of MDPs. We reformulate the problem as a two-player stochastic game and propose an approximate algorithm based on Monte Carlo tree search and Bayesian optimisation. Our experiments demonstrate that our approach significantly outperforms baseline approaches for this problem.

1 Introduction

In standard model-based reinforcement learning (RL), an agent interacts with an unknown environment to maximise the expected reward [42]. However, for any given episode the total reward received by the agent is uncertain. There are two sources of this uncertainty: the *epistemic* (or *parametric*) uncertainty due to imperfect knowledge about the underlying MDP model, and *aleatoric* (or *internal*) uncertainty due to the inherent stochasticity of the underlying MDP [20]. In many domains, we wish to find policies which are risk-averse to both sources of uncertainty.

As an illustrative example, consider a navigation system for an automated taxi service. The system attempts to minimise travel duration subject to uncertainty due to the traffic conditions, and traffic lights and pedestrian crossings. For each new day of operation, the traffic conditions are initially unknown. This corresponds to epistemic uncertainty over the MDP model. This uncertainty is reduced as the agent collects experience and learns which roads are busy or not busy. Traffic lights and pedestrian crossings cause stochasticity in the transition durations corresponding to aleatoric uncertainty. The aleatoric uncertainty remains even as the agent learns about the environment. A direct route, i.e. through the centre of the city, optimises the expected journey duration. However, occasionally this route incurs extremely long durations due to some combination of poor traffic conditions and getting stuck at traffic lights and pedestrian crossings. Thus, bad outcomes can be caused by a combination of both sources of uncertainty. These rare outcomes may result in unhappy customers for the taxi service. Therefore, we wish to be risk-averse and prioritise greater certainty of avoiding poor outcomes rather than expected performance. To ensure that the risk of a bad journey is avoided, the navigation system must consider both the epistemic and the aleatoric uncertainties.

In model-based Bayesian RL, a belief distribution over the underlying MDP is maintained. This quantifies the epistemic uncertainty over the underlying MDP given the transitions observed so far.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

The Bayesian RL problem can be reformulated as a planning problem in a *Bayes-Adaptive* MDP (BAMDP) with an augmented state space composed of the belief over the underlying MDP and the state in the underlying MDP [12].

In this work, we address risk-averse decision making in model-based Bayesian RL. Instead of optimising for expected value, we optimise a risk metric applied to the total return of the BAMDP. Specifically, we focus on *conditional value at risk* (CVaR) [34]. The return in a BAMDP is uncertain due to both the uncertain prior belief over the underlying MDP, and the inherent stochasticity of MDPs. By optimising the CVaR of the return in the BAMDP, our approach simultaneously addresses epistemic *and* aleatoric uncertainty under a single framework. This is in contrast with previous works which have generally considered risk-aversion to either epistemic *or* aleatoric uncertainty.

We formulate CVaR optimisation in Bayesian RL as a stochastic game over the BAMDP against an adversarial environment. Solving this game is computationally challenging because the set of reachable augmented states grows exponentially, and the action space of the adversary is continuous. Our proposed algorithm uses Monte Carlo tree search (MCTS) to focus search on promising areas of the augmented state space. To deal with the continuous action space of the adversary, we use Bayesian optimisation to expand promising adversary actions. Our main contributions are:

- Addressing CVaR optimisation of the return in model-based Bayesian RL to simultaneously mitigate epistemic and aleatoric uncertainty.
- An algorithm based on MCTS and Bayesian optimisation for this problem.

To our knowledge, this is the first work to optimise a risk metric in model-based Bayesian RL to simultaneously address both epistemic and aleatoric uncertainty, and the first work to present an MCTS algorithm for CVaR optimisation in sequential decision making problems. Our empirical results show that our algorithm significantly outperforms baseline methods on two domains.

2 Related Work

Many existing works address aversion to risk stemming from either epistemic model uncertainty or aleatoric uncertainty. One of the most popular frameworks for addressing epistemic uncertainty is the Robust or Uncertain MDP [17, 26, 33, 47] which considers worst-case expected value over a set of possible MDPs. Other works address epistemic uncertainty by assuming that a distribution over the underlying MDP is known. One approach is to find a policy which optimises the expected value in the worst k% of parameter realisations [5, 11]. Most related to our work is that of Sharma et al. [38] which optimises a risk metric in the Bayesian RL context where the prior is a discrete distribution over a finite number of MDPs. However, all of these existing works marginalise out the aleatoric uncertainty by considering the expected value under each possible MDP. In contrast, we optimise CVaR of the return in the Bayesian RL setting to address both epistemic and aleatoric uncertainty.

Conditional value at risk (CVaR) [34] is a common coherent risk metric which has been applied to MDPs to address aleatoric uncertainty. For an introduction to coherent risk metrics, see [2]. In this paper, we consider *static* risk, where the risk metric is applied to the total return, rather than *dynamic* risk, where the risk metric is applied recursively [43]. For this static CVaR setting, approaches based on value iteration over an augmented state space have been proposed [3, 8, 32]. Other works propose policy gradient methods to find a locally optimal solution for the optimisation of CVaR [4, 6, 7, 31, 43, 45, 46], or general coherent risk metrics [44]. These existing approaches all assume the agent has access to the true underlying MDP for training/planning, and optimise the CVaR in that single MDP. In our work, we address the model-based Bayesian RL setting where we only have access to a prior belief distribution over MDPs. We find Bayes-adaptive policies which are risk-averse to both the epistemic uncertainty over the underlying MDP and the aleatoric uncertainty due to stochastic transitions. To our knowledge, no existing work has addressed CVaR optimisation in the Bayes-adaptive RL setting to simultaneously address both forms of uncertainty.

Constrained MDPs impose a constraint on the expected value of a cost [1, 36]. Such constraints have been addressed in Bayesian RL [18]. However, this approach only constrains the expected cost and it is unclear how to choose cost penalties to obtain the desired risk averse behaviour. Monte Carlo Tree Search (MCTS) has been adapted to expected cost constraints [19]. However, to our knowledge, this is the first work to adapt MCTS to CVaR optimisation in sequential decision making problems.

In the Deep RL setting, Deep Q-Networks [24] have been modified to predict the uncertainty over the Q-values associated with epistemic and aleatoric uncertainty [9, 13] and derive a risk-sensitive policy. In this work, we propose to use the Bayes-adaptive formulation of RL to model and address both uncertainty sources. Another related strand of work is robust adversarial reinforcement learning [28, 30] (RARL) which optimises robust policies by simultaneously training an adversary which perturbs the environment. In RARL it is unclear how much power to give to the adversary to achieve the desired level of robustness. Our approach ensures that the CVaR at the desired confidence level is optimised.

3 Preliminaries

Let Z be a bounded-mean random variable, i.e. $E[|Z|] < \infty$, on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, with cumulative distribution function $F(z) = \mathcal{P}(Z \leq z)$. In this paper we interpret Z as the total reward, or return, to be maximised. The *value at risk* at confidence level $\alpha \in (0, 1]$ is defined as $\text{VaR}_\alpha(Z) = \min\{z|F(z) \geq \alpha\}$. The *conditional value at risk* at confidence level α is defined as

$$\text{CVaR}_\alpha(Z) = \frac{1}{\alpha}\int_0^\alpha \text{VaR}_\gamma(Z)d\gamma. \tag{1}$$

If Z has a continuous distribution, $\text{CVaR}_\alpha(Z)$ can be defined using the more intuitive expression: $\text{CVaR}_\alpha(Z) = \mathbb{E}[Z|Z \leq \text{VaR}_\alpha(Z)]$. Thus, $\text{CVaR}_\alpha(Z)$ may be interpreted as the expected value of the α-portion of the left tail of the distribution of Z. CVaR may also be defined as the expected value under a perturbed distribution using its dual representation [34, 37]

$$\text{CVaR}_\alpha(Z) = \min_{\xi \in \mathcal{B}(\alpha,\mathcal{P})} \mathbb{E}_\xi\big[Z\big], \tag{2}$$

where $\mathbb{E}_\xi[Z]$ denotes the ξ-weighted expectation of Z, and the *risk envelope*, $\mathcal{B}$, is given by

$$\mathcal{B}(\alpha, \mathcal{P}) = \left\{\xi : \xi(\omega) \in \left[0, \frac{1}{\alpha}\right], \int_{\omega\in\Omega} \xi(\omega)\mathcal{P}(\omega)d\omega = 1\right\}. \tag{3}$$

Therefore, the CVaR of a random variable Z may be interpreted as the expectation of Z under a worst-case perturbed distribution, $\xi\mathcal{P}$. The risk envelope is defined so that the probability density of any outcome can be increased by a factor of at most $1/\alpha$, whilst ensuring the perturbed distribution is a valid probability distribution.

3.1 CVaR Optimisation in Standard MDPs

A finite-horizon Markov decision process (MDP) is a tuple, $\mathcal{M} = (S, A, R, T, H, s_0)$, where S and A are finite state and action spaces, $R : S \times A \to \mathbb{R}$ is the reward function, $T : S \times A \times S \to [0, 1]$ is the transition function, H is the horizon, and s_0 is the initial state. A history of $\mathcal{M}$ is a sequence $h = s_0a_0s_1a_1 \ldots a_{t-1}s_t$ such that $T(s_i, a_i, s_{i+1}) > 0$ for all i. We denote the set of all histories over $\mathcal{M}$ as $\mathcal{H}^\mathcal{M}$. A history-dependent policy is a function $\pi : \mathcal{H}^\mathcal{M} \to A$, and we write $\Pi^\mathcal{M}_\mathcal{H}$ to denote the set of all history-dependent policies for $\mathcal{M}$. If π only depends on the last state s_t of h, then we say π is *Markovian*, and we denote the set of all Markovian policies as $\Pi^\mathcal{M}$. A policy π induces a distribution $\mathcal{P}^\mathcal{M}_\pi$ over histories and we define $G^\mathcal{M}_\pi = \sum_{t=0}^H R(s_t, a_t)$ as the distribution over total returns of $\mathcal{M}$ under π. Many existing works have addressed the problem of optimising the static CVaR of $G^\mathcal{M}_\pi$, defined as follows.

Problem 1 (CVaR optimisation in standard MDPs) *Let $\mathcal{M}$ be an MDP. Find the optimal CVaR of the total return at confidence level α:*

$$\max_{\pi\in\Pi^\mathcal{M}_\mathcal{H}} \text{CVaR}_\alpha(G^\mathcal{M}_\pi). \tag{4}$$

Unlike dynamic Markov risk measures [35], history-dependent policies may be required to optimally solve this static CVaR optimisation problem [3, 8, 43]. Previous works [8, 27] have shown that this

problem can be interpreted either as being risk-averse to aleatoric uncertainty, *or* as being robust to epistemic uncertainty. The latter interpretation comes from the dual representation of CVaR (Eq. 2), where CVaR is expressed as the expected value under a worst-case perturbed probability distribution.

Methods based on dynamic programming have been proposed to solve Problem 1 [3, 8, 29]. In particular, Chow et al. [8] augment the state space by an additional continuous state factor, $y \in (0, 1]$, corresponding to the confidence level. The value function for the augmented state (s, y) is defined by $V(s, y) = \max_{\pi \in \Pi^{\mathcal{M}}_{\mathcal{H}}} \text{CVaR}_y(G^{\mathcal{M}}_{\pi})$, and can be computed using the following Bellman equation

$$V(s, y) = \max_{a \in A} \left[R(s, a) + \min_{\xi \in \mathcal{B}(y, T(s,a,\cdot))} \sum_{s' \in S} \xi(s') V(s', y\xi(s')) T(s, a, s') \right], \tag{5}$$

where ξ represents an adversarial perturbation to the transition probabilities, and the additional state variable, y, ensures that the perturbation to the probability of any history through the MDP is at most $1/y$. Thus, y can be thought of as keeping track of the "budget" to adversarially perturb the probabilities. Therefore, $V(s_0, \alpha)$ is the expected value under the adversarial probability distribution defined by Eq. 2 and 3 and corresponds to the optimal CVaR in the MDP (Problem 1). To address the continuous state variable, y, [8] use dynamic programming with linear function approximation.

3.2 Stochastic Games

In this paper, we will formulate CVaR optimisation as a turn-based two-player zero-sum stochastic game (SG). An SG between an agent and an adversary is a generalisation of an MDP and can be defined using a similar tuple $\mathcal{G} = (S, A, T, R, H, s_0)$. The elements of $\mathcal{G}$ are interpreted as with MDPs, but extra structure is added. In particular, S is partitioned into a set of agent states S^{agt}, and a set of adversary states S^{adv}. Similarly, A is partitioned into a set of agent actions A^{agt}, and a set of adversary actions A^{adv}. The transition function is defined such that agent actions can only be executed in agent states, and adversary actions can only be executed in adversary states.

We denote the set of Markovian agent policies mapping agent states to agent actions as $\Pi^{\mathcal{G}}$ and the set of Markovian adversary policies, defined similarly, as $\Sigma^{\mathcal{G}}$. A pair (π, σ) of agent-adversary policies induces a probability distribution over histories and we define $G^{\mathcal{G}}_{(\pi,\sigma)} = \sum_{t=0}^{H} R(s_t, a_t)$ as the distribution over total returns of $\mathcal{G}$ under π and σ. In a zero-sum SG, the agent seeks to maximise the expected return reward, whilst the adversary seeks to minimise it:

$$\max_{\pi \in \Pi^{\mathcal{G}}} \min_{\sigma \in \Sigma^{\mathcal{G}}} \mathbb{E}\big[G^{\mathcal{G}}_{(\pi,\sigma)}\big]. \tag{6}$$

3.3 Bayesian RL

Here we describe the Bayesian formulation of decision making in an unknown MDP [21, 12]. In Bayesian RL, the dynamics of the MDP $\mathcal{M}$ are unknown and we assume that its transition function, T, is a latent variable distributed according to a prior distribution $\mathcal{P}(T|h_0)$, where h_0 represents the empty history where no transitions have been observed. After observing a history $h = s_0 a_0 s_1 a_1 \dots a_{t-1} s_t$, the posterior belief over T is updated using Bayes' rule: $\mathcal{P}(T|h) \propto \mathcal{P}(h|T)\mathcal{P}(T|h_0)$.

In the Bayes-Adaptive MDP (BAMDP) formulation of Bayesian RL [12], the uncertainty about the model dynamics is captured by augmenting the state space to include the history: $S^+ = S \times \mathcal{H}$. This captures the model uncertainty as the history is a sufficient statistic for the posterior belief over T given the initial prior belief. The transition and reward functions for this augmented state space are

$$T^+((s, h), a, (s', has')) = \int_T T(s, a, s') \mathcal{P}(T|h) dT, \tag{7}$$

$$R^+((s, h), a) = R(s, a). \tag{8}$$

The initial augmented state is $s^+_0 = (s_0, h_0)$. The tuple $\mathcal{M}^+ = (S^+, A, T^+, R^+, H, s^+_0)$ defines the BAMDP. Typically, solutions to the Bayesian RL problem attempt to (approximately) find a

Bayes-optimal policy, which maximises the expected sum of rewards under the prior belief over the transition function. In this work, we address risk-averse solutions to the Bayesian RL problem.

4 CVaR Optimisation in Bayes-Adaptive RL

Many existing works address CVaR optimisation in standard MDPs. Here, we assume that we only have access to a prior distribution over MDPs, and optimise the CVaR in the corresponding BAMDP.

Problem 2 (CVaR optimisation in Bayes-Adaptive MDPs) *Let $\mathcal{M}^+$ be a Bayes-Adaptive MDP. Find the optimal CVaR at confidence level α:*

$$\max_{\pi \in \Pi^{\mathcal{M}^+}} \text{CVaR}_\alpha(G_\pi^{\mathcal{M}^+}). \tag{9}$$

Posing CVaR optimisation over BAMDPs leads to two observations. First, unlike Problem 1, we do not need to consider history-dependent policies as the history is already embedded in the BAMDP state. Second, the solution to this problem corresponds to a policy which is risk-averse to both the epistemic uncertainty due to the uncertain belief over the MDP, and the aleatoric uncertainty due to the stochasticity of the underlying MDP. In the following, we elaborate on this second observation.

4.1 Motivation: Robustness to Epistemic and Aleatoric Uncertainty

The solution to Problem 2 is risk-averse to both the epistemic *and* the aleatoric uncertainty. Intuitively, this is because the return distribution in the BAMDP is influenced both by the prior distribution over models as well as stochastic transitions. In this section, we introduce a novel alternative definition of CVaR in BAMDPs to formally illustrate this concept. Specifically, we show that CVaR in a BAMDP is equivalent to the expected value under an adversarial prior distribution over transition functions, and adversarial perturbations to the transition probabilities in all possible transition functions. The perturbation "budget" of the adversary is split multiplicatively between perturbing the probability density of transition functions, and perturbing the transition probabilities along any history. Therefore, the adversary has the power to minimise the return for the agent through a combination of modifying the prior so that "bad" transition functions are more likely (epistemic uncertainty), and making "bad" transitions within any given transition function more likely (aleatoric uncertainty).

We consider an adversarial setting, whereby an adversary is allowed to apply perturbations to modify the prior distribution over the transition function as well as the transition probabilities within each possible transition function, subject to budget constraints. Define $\mathcal{T}$ to be the support of $\mathcal{P}(T|h_0)$, and let $T_p \in \mathcal{T}$ be a plausible transition function. Note that, in general, $\mathcal{T}$ can be a continuous set. Consider a perturbation of the prior distribution over transition functions $\delta : \mathcal{T} \to \mathbb{R}_{\geq 0}$ such that $\int_{T_p} \delta(T_p)\mathcal{P}(T_p|h_0)dT_p = 1$. Additionally, for $T_p \in \mathcal{T}$, consider a perturbation of T_p, $\xi_p : S \times A \times S \to \mathbb{R}_{\geq 0}$ such that $\sum_{s' \in S} \xi_p(s,a,s') \cdot T_p(s,a,s') = 1 \ \ \forall s, a$. We denote the set of all possible perturbations of T_p as Ξ_p, and the set of all perturbations over $\mathcal{T}$ as $\Xi = \times_{T_p \in \mathcal{T}} \Xi_p$. For BAMDP $\mathcal{M}^+$, $\delta : \mathcal{T} \to \mathbb{R}_{\geq 0}$, and $\xi \in \Xi$, we define $\mathcal{M}^+_{\delta,\xi}$ as the BAMDP, obtained from $\mathcal{M}^+$, with perturbed prior distribution $\mathcal{P}^\delta(T_p|h_0) = \delta(T_p) \cdot \mathcal{P}(T_p|h_0)$ and perturbed transition functions $T_p^\xi(s,a,s') = \xi_p(s,a,s') \cdot T_p(s,a,s')$. We are now ready to state Prop. 1, which provides an interpretation of CVaR in BAMDPs as expected value under an adversarial prior distribution and adversarial transition probabilities. All propositions are proven in the supplementary material.

Proposition 1 (CVaR in BAMDPs) *Let $\mathcal{M}^+$ be a BAMDP. Then:*

$$\text{CVaR}_\alpha(G_\pi^{\mathcal{M}^+}) = \min_{\substack{\delta : \mathcal{T} \to \mathbb{R}_{\geq 0} \\ \xi \in \Xi}} \mathbb{E}\big[G_\pi^{\mathcal{M}^+_{\delta,\xi}}\big], \quad \textbf{\textit{s.t.}} \tag{10}$$

$$0 \leq \delta(T_p)\xi_p(s_0,a_0,s_1)\xi_p(s_1,a_1,s_2)\cdots\xi_p(s_{H\text{-}1},a_{H\text{-}1},s_H) \leq \frac{1}{\alpha},$$
$$\forall\, T_p \in \mathcal{T},\ \forall (s_0,a_0,s_1,a_1,\ldots,s_H) \in \mathcal{H}_H.$$

Prop. 1 shows that CVaR can be computed as expected value under an adversarial prior distribution over transition functions, and adversarial perturbations to the transition probabilities in all possible transition functions. Therefore, a policy which optimises the CVaR in a BAMDP must mitigate the risks due to both sources of uncertainty.

4.2 Bayes-Adaptive Stochastic Game Formulation

In Prop. 1, we formulated CVaR in BAMDPs as expected value under adversarial perturbations to the prior, and to each possible MDP. However, this formulation is difficult to solve directly as it requires optimising over the set of variables Ξ, which can have infinite dimension since the space of plausible transition functions may be infinite. Thus, we formulate CVaR optimisation in BAMDPs in a way that is more amenable to optimisation: an SG played on an augmented version of the BAMDP. The agent plays against an adversary which modifies the transition probabilities of the original BAMDP. Perturbing the BAMDP transition probabilities is equivalent to perturbing both the prior distribution and each transition function as considered by Prop. 1. This extends work on CVaR optimisation in standard MDPs [8] to the Bayes-adaptive setting.

We begin by augmenting the state space of the BAMDP with an additional state factor, $y \in (0, 1]$, so that now the augmented state comprises $(s, h, y) \in S^+_{\mathcal{G}} = S \times \mathcal{H} \times (0, 1]$. The stochastic game proceeds as follows. Given an augmented state, $(s, h, y) \in S^{+,agt}_{\mathcal{G}}$, the agent applies an action, $a \in A$, and receives reward $R^+_{\mathcal{G}}((s, h, y), a) = R(s, a)$. The state then transitions to an adversary state $(s, ha, y) \in S^{+,adv}_{\mathcal{G}}$. The adversary then chooses an action from a continuous action space, $\xi \in \Xi(s, a, h, y)$, to perturb the BAMDP transition probabilities where

$$\Xi(s, a, h, y) = \Big\{\xi : S \to \mathbb{R}_{\geq 0} \;\Big|\; 0 \leq \xi(s') \leq 1/y \;\; \forall s' \text{ and } \textstyle\sum_{s' \in S} \xi(s') \cdot T^+((s, h), a, (s', has')) = 1\Big\}. \tag{11}$$

In Eq. 11, the adversary perturbation is restricted to adhere to the budget defined by Eq. 3 so that the probability of any history in the BAMDP is perturbed by at most $1/y$. After the adversary chooses the perturbation action, the game transitions back to an agent state $(s', has', y \cdot \xi(s')) \in S^{+,agt}_{\mathcal{G}}$ according to the transition function

$$T^+_{\mathcal{G}}((s, ha, y), \xi, (s', has', y \cdot \xi(s'))) = \xi(s') \cdot T^+((s, h), a, (s', has')). \tag{12}$$

The initial augmented state of the stochastic game is $s^+_{0,\mathcal{G}} = (s_0, h_0, \alpha) \in S^{+,agt}_{\mathcal{G}}$. The tuple $\mathcal{G}^+ = (S^+_{\mathcal{G}}, A^+_{\mathcal{G}}, T^+_{\mathcal{G}}, R^+_{\mathcal{G}}, H, s^+_{0,\mathcal{G}})$ defines the Bayes-Adaptive CVaR Stochastic Game (BA-CVaR-SG), where the joint action space is $A^+_{\mathcal{G}} = A \cup \Xi$.

The maximin expected value equilibrium of the BA-CVaR-SG optimises the expected value of the BAMDP under perturbations to the probability distribution over paths by an adversary subject to the constraint defined by Eq. 3. Therefore, Prop. 2 states that the maximin expected value equilibrium in the BA-CVaR-SG corresponds to the solution to CVaR optimisation in BAMDPs (Problem 2).

Proposition 2 *Formulation of CVaR optimisation in BAMDPs as a stochastic game.*

$$\max_{\pi \in \Pi^{\mathcal{M}^+}} \text{CVaR}_\alpha(G^{\mathcal{M}^+}_\pi) = \max_{\pi \in \Pi^{\mathcal{G}^+}} \min_{\sigma \in \Sigma^{\mathcal{G}^+}} \mathbb{E}\big[G^{\mathcal{G}^+}_{(\pi,\sigma)}\big]. \tag{13}$$

Prop. 2 directly extends Prop. 1 from [8] to BAMDPs. The BAMDP transition function is determined by both the prior distribution over transition functions and the transition functions themselves (Eq. 7). Therefore, perturbing the BAMDP transition function, as considered by Prop. 2, is equivalent to perturbing both the prior distribution and each transition function as considered by Prop. 1.

4.3 Monte Carlo Tree Search Algorithm

Solving the BA-CVaR-SG is still computationally difficult due to two primary reasons. First, the set of reachable augmented states grows exponentially with the horizon due to the exponentially growing set of possible histories. This prohibits the use of the value iteration approach proposed by Chow et al. [8] for standard MDPs. Second, CVaR is considerably more difficult to optimise than expected value. In the BA-CVaR-SG formulation, this difficulty is manifested in the continuous action space of the adversary corresponding to possible perturbations to the BAMDP transition function.

In this section we present an approximate algorithm, Risk-Averse Bayes-Adaptive Monte Carlo Planning (RA-BAMCP). Like BAMCP [16], which applies MCTS to BAMDPs, and applications of

Algorithm 1: Risk-Averse Bayes-Adaptive Monte Carlo Planning

function `Search`($\mathcal{G}^+$)
- create root node v_0 with:
  - $s^+_{\mathcal{G}}(v_0) = s^+_{0,\mathcal{G}}$
  - $player(v_0) = agent$
- **while** within computational budget **:**
  - $v_{leaf} \leftarrow$ `TreePolicy`(v_0)
  - $\widetilde{r} \leftarrow$ `DefaultPolicy`(v_{leaf})
  - `UpdateNodes`($v_{leaf}, \widetilde{r}$)
- $v' = \underset{v' \in \text{children of } v_0}{\arg\max} \ \widehat{Q}(v')$
- $v'' = \underset{v'' \in \text{children of } v'}{\arg\min} \ \widehat{Q}(v'')$
- **return** $a(v'), \xi(v'')$

function `TreePolicy`(v)
- **while** v is non-terminal **:**
  - **if** $player(v) = agent$ **:**
    - **if** v not fully expanded **:**
      - **return** `ExpandAgent`(v)
    - **else**
      - $v \leftarrow v$.`BestChild()`
  - **else if** $player(v) = adv$ **:**
    - **if** $(N(v))^\tau \geq |\widetilde{\Xi}(v)|$ **:**
      - **return** `ExpandAdversary`(v)
    - **else**
      - $v \leftarrow v$.`BestChild()`
  - **else if** $player(v) = chance$ **:**
    - $v \leftarrow$ sample successor according to Eq. 12

function `UpdateNodes`(v, $\widetilde{r}$)
- **while** v is not null **:**
  - $N(v) += 1$
  - $\widehat{Q}(v) += \frac{\widetilde{r}(v) - \widehat{Q}(v)}{N(v)}$
  - $v \leftarrow$ parent of v

function `ExpandAgent`(v)
- choose $a \in$ untried actions from action set A
- add child v' to v with:
  - $s^+_{\mathcal{G}}(v') = s^+_{\mathcal{G}}(v)$
  - $a(v') = a$
  - $player(v') = adv$
  - $\widetilde{\Xi}(v') = \varnothing$
- **return** v'

function `ExpandAdversary`(v)
- **if** $\widetilde{\Xi}(v) = \varnothing$ **:**
  - $\xi_{new} =$ `RandomPerturbation`(v)
- **else**
  - $\xi_{new} =$ `BayesOptPerturbation`(v)
- $\widetilde{\Xi}(v)$.insert(ξ_{new})
- add child v' to v with:
  - $s^+_{\mathcal{G}}(v') = s^+_{\mathcal{G}}(v)$
  - $\xi(v') = \xi$
  - $player(v') = chance$
- **return** v'

function `BestChild`(v)
- **if** *player(v) = agent* **:**
  - **return** $\underset{v' \in \text{children of } v}{\arg\max} \left[\widehat{Q}(v') + c_{mcts} \cdot \sqrt{\frac{\log N(v)}{N(v')}}\right]$
- **else**
  - **return** $\underset{v' \in \text{children of } v}{\arg\min} \left[\widehat{Q}(v') - c_{mcts} \cdot \sqrt{\frac{\log N(v)}{N(v')}}\right]$

MCTS to two-player games such as Go [14], we use MCTS to handle the large state space by focusing search in promising areas. Unlike BAMCP, we perform search over a Bayes-adaptive stochastic game against an adversary which may perturb the probabilities, and we cannot use root sampling as we need access to the BAMDP transition dynamics (Eq. 7) to compute the set of admissible adversary actions (Eq. 11). To handle the continuous action space for the adversary, we use progressive widening [10] paired with Bayesian optimisation [25, 22] to prioritise promising actions.

RA-BAMCP Outline Algorithm 1 proceeds using the standard components of MCTS: selecting nodes within the tree, expanding leaf nodes, performing rollouts using a default policy, and updating the statistics in the tree. The search tree consists of alternating layers of three types of nodes: agent decision nodes, adversary decision nodes, and chance nodes. At each new node v, the value estimate associated with that node, $\widehat{Q}(v)$, and visitation count, $N(v)$, are both initialised to zero. At agent (adversary) decision nodes we use an upper (lower) confidence bound to optimistically select actions within the tree. At newly expanded chance nodes we compute $\mathcal{P}(T|h)$, the posterior distribution over the transition function for the history leading up to that chance node. Using this posterior distribution, we can compute the transition function for the BA-CVaR-SG, $T^+_{\mathcal{G}}$, at that chance node. When a chance node is reached during a simulation, the successor state is sampled using $T^+_{\mathcal{G}}$.

At each adversary node, a discrete set of perturbation actions is available to the adversary, denoted by $\widetilde{\Xi}(v)$. We use progressive widening [10] to gradually increase the set of perturbation actions available by adding actions from the continuous set of possible actions, $\Xi(v)$. The rate at which new actions are expanded is controlled by the progressive widening parameter, τ. A naive implementation of

progressive widening would simply sample new perturbation actions randomly. This might require many expansions before finding a good action. To mitigate this issue we use Gaussian process (GP) Bayesian optimisation to select promising actions to expand in the tree [25, 22].

In our algorithm the `BayesOptPerturbation` function performs Bayesian optimisation to return a promising perturbation action at a given adversary node, v. We train a GP using a Gaussian kernel. The input features are existing expanded perturbation actions at the node, $\xi \in \widetilde{\Xi}(v)$. The number of input dimensions is equal to the number of successor states with non-zero probability according to the BAMDP transition function. The training labels are the corresponding value estimates in the tree for each of the perturbation actions: $\widehat{Q}(v)$, for all children v' of v. The acquisition function for choosing the new perturbation action to expand is a lower confidence bound which is optimistic for the minimising adversary: $\xi_{new} = \arg\min_{\xi}[\mu(\xi) - c_{bo} \cdot \sigma(\xi)]$, where $\mu(\xi)$ and $\sigma(\xi)$ are the mean and standard deviations of the GP predicted posterior distribution respectively. Prop. 3 establishes that a near-optimal perturbation will eventually be expanded with high probability if c_{bo} is set appropriately.

Proposition 3 *Let $\delta \in (0, 1)$. Provided that c_{bo} is chosen appropriately (details in the appendix), as the number of perturbations expanded approaches ∞, a perturbation within any $\epsilon > 0$ of the optimal perturbation will be expanded by the Bayesian optimisation procedure with probability $\geq 1 - \delta$.*

After each simulation, the value estimates at each node are updated using $\widetilde{r}(v)$, the reward received from v onwards during the simulation. Search continues by performing simulations until the computational budget is exhausted. At this point, the estimated best action for the agent and the adversary is returned. In our experiments we perform online planning: after executing an action and transitioning to a new state, we perform more simulations from the new root state.

5 Experiments

Code for the experiments is included in the supplementary material. All algorithms are implemented in C++ and Gurobi is used to solve linear programs for the value iteration (VI) approaches. Computation times are reported for a 3.2 GHz Intel i7 processor with 64 GB of RAM.

No existing works address the problem formulation in this paper. Therefore, we adapt two methods for CVaR optimisation to the BAMDP setting and also compare against approaches for related problems to assess their performance in our setting. We compare the following approaches:

- *RA-BAMCP*: the approach presented in this paper.
- *CVaR BAMDP Policy Gradient* (PG): applies the policy gradient approach to CVaR optimisation from [45] with the vectorised belief representation for BAMDPs proposed in [15] with linear function approximation. Further details on this approach can be found in the supplementary material.
- *CVaR VI BAMDP*: the approximate VI CVaR optimisation approach from [8] applied to the BAMDP. Due to the extremely poor scalability of this approach we only test it on the smaller of our domains.
- *CVaR VI Expected MDP* (EMDP): the approximate VI approach from [8] applied to the expected MDP parameters according to the prior distribution over transition functions, $\mathcal{P}(T|h_0)$.
- *BAMCP*: Bayes-Adaptive Monte Carlo Planning which optimises the expected value [16]. This is equivalent to *RA-BAMCP* with the confidence level, α, set to 1.

For *RA-BAMCP* and *BAMCP* the MCTS exploration parameter was set to $c_{mcts} = 2$ based on empirical performance. For these methods, we performed $100,000$ simulations for the initial step and $25,000$ simulations per step thereafter. For both of these algorithms the rollout policy used for the agent was the *CVaR VI Expected MDP* policy. For *RA-BAMCP*, the rollout policy for the adversary was random, the progressive widening parameter was set to $\tau = 0.2$, and the exploration parameter for the GP Bayesian optimisation was also set to $c_{bo} = 2$. Details of the GP hyperparameters are in the supplementary material. For *CVaR VI Expected MDP* and *CVaR VI BAMDP*, 20 log-spaced points were used to discretise the continuous state space for approximate VI as described by [8]. For *CVaR BAMDP Policy Gradient* we trained using 2×10^6 simulations and details on hyperparameter tuning are in the supplementary material.

Betting Game Domain We adapt this domain from the literature on conditional value at risk in Markov decision processes [3] to the Bayes-adaptive setting. The agent begins with $money = 10$. At each stage the agent may choose to place a bet from $bets = \{0, 1, 2, 5, 10\}$ provided that sufficient money is available. If the agent wins the game at that stage, an amount of money equal to the bet placed is received. If the agent loses the stage, the amount of money bet is lost. The reward is the total money the agent has after 6 stages of betting. In our Bayes-adaptive setting, the probability of winning the game is not known apriori. Instead, a prior distribution over the probability of winning the game is modelled using a beta distribution with parameters $\alpha = \frac{10}{11}, \beta = \frac{1}{11}$, indicating that the odds for the game are likely to be in favour of the agent.

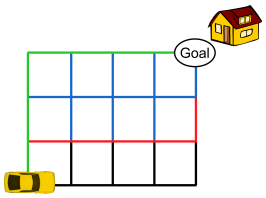


Figure 1: Autonomous car navigation domain. Colours indicate the road types as follows - black: *highway*, red: *main road*, blue: *street*, green: *lane*.

Autonomous Car Navigation Domain An autonomous car must navigate along roads to a destination by choosing from actions {*up, down, left, right*} (Figure 1). There are four types of roads: {*highway, main road, street, lane*} with different transition duration characteristics. The transition distribution for each type of road is unknown. Instead, the prior belief over the transition duration distribution for each road type is modelled by a Dirichlet distribution. Thus, the belief is represented by four Dirichlet distributions. Details on the Dirichlet prior used for each road type are included in the supplementary material. Upon traversing each section of road, the agent receives reward $-time$ where $time$ is the transition duration for traversing that section of road. Upon reaching the destination, the agent receives a reward of 80. The search horizon was set to $H = 10$.

5.1 Results

We evaluated the return from each method over 2000 episodes. For each episode, the true underlying MDP was sampled from the prior distribution. Therefore, the return received for each evaluation episode is subject to both epistemic uncertainty due to the uncertain MDP model, and aleatoric uncertainty due to stochastic transitions. Results for this evaluation can be found in Table 1. The rows indicate the method and the confidence level, α, that the method is set to optimise. The columns indicate the performance of each method for CVaR with $\alpha = 0.03$ and $\alpha = 0.2$, and expected value ($\alpha = 1$), evaluated using the sample of 2000 episodes. Note that when evaluating the performance of each algorithm, we only care about the CVaR performance for the confidence level that the algorithm is set to optimise.

In the Betting Game domain, strong performance is attained by *CVaR VI BAMDP*, however the computation time required for this approach is considerable. It was not possible to test *CVaR VI BAMDP* on the Autonomous Navigation due to its poor scalability. *RA-BAMCP* attained near-optimal performance at both confidence levels using substantially less computation. *CVaR VI Expected MDP* performed well at the $\alpha = 0.2$ confidence level, but very poorly at $\alpha = 0.03$. This approach performs poorly at $\alpha = 0.03$ because by using the expected value of the transition function, the probability of repeatedly losing the game is underestimated. *BAMCP* performed the best at optimising expected value, but less well than other approaches for optimising CVaR.

CVaR BAMDP Policy Gradient performed strongly in the betting domain with $\alpha = 0.2$. However, it attained poor asymptotic performance with $\alpha = 0.03$. Conversely, this method performed well in the navigation domain with $\alpha = 0.03$, but poorly for $\alpha = 0.2$. These results indicate that the policy gradient approach is susceptible to local minima. Training curves for *CVaR BAMDP Policy Gradient* can be found in the supplementary material.

In the Autonomous Car Navigation domain *RA-BAMCP* outperformed *CVaR VI Expected MDP* at both confidence levels. As expected, *BAMCP* again performed the best at optimising expected value, but not CVaR. The return histograms in Figure 2 show that while *BAMCP* optimises expected value, there is high uncertainty over the return value. As the confidence level α is decreased, applying *RA-BAMCP* decreases the likelihood of poor returns by shifting the left tail of the return distribution to the right, at the expense of worse expected performance. This demonstrates that *RA-BAMCP* is risk-averse to poor return values in the presence of both epistemic and aleatoric uncertainty.

Method	Time (s)	CVaR (α=0.03)	CVaR (α=0.2)	Expected Value
RA-BAMCP (α=0.03)	17.9 (0.2)	9.98 (0.02)	10.00 (0.002)	10.00 (0.001)
CVaR VI BAMDP (α=0.03)	8620.5	**10.00** (0.0)	10.00 (0.0)	10.00 (0.0)
CVaR VI EMDP (α=0.03)	61.7	0.00 (0.0)	10.45 (0.33)	15.72 (0.09)
CVaR BAMDP PG (α=0.03)	4639.3	2.90 (0.06)	10.57 (0.33)	27.34 (0.24)
RA-BAMCP (α=0.2)	78.9 (0.2)	0.00 (0.0)	20.09 (1.04)	46.84 (0.37)
CVaR VI BAMDP (α=0.2)	8620.5	0.00 (0.0)	**20.77** (1.02)	44.15 (0.33)
CVaR VI EMDP (α=0.2)	61.7	0.00 (0.0)	19.33 (0.99)	39.86 (0.30)
CVaR BAMDP PG (α=0.2)	4309.7	0.00 (0.0)	19.85 (0.97)	46.65 (0.37)
BAMCP	13.5 (0.01)	0.00 (0.0)	17.07 (1.09)	**59.36** (0.52)

(a) Betting Game domain

Method	Time (s)	CVaR (α=0.03)	CVaR (α=0.2)	Expected Value
RA-BAMCP (α=0.03)	255.7 (0.2)	23.51 (0.17)	25.41 (0.06)	29.56 (0.06)
CVaR VI EMDP (α=0.03)	96.6	20.65 (0.68)	26.11 (0.16)	30.13 (0.07)
CVaR BAMDP PG (α=0.03)	37232.6	**23.92** (0.08)	25.38 (0.05)	28.97 (0.05)
RA-BAMCP (α=0.2)	205.1 (0.3)	18.91 (0.44)	**27.76** (0.24)	38.33 (0.15)
CVaR VI EMDP (α=0.2)	96.6	4.20 (0.54)	20.42 (0.44)	36.64 (0.21)
CVaR BAMDP PG (α=0.2)	36686.1	10.05 (0.74)	24.70 (0.39)	49.67 (0.36)
BAMCP	74.4 (0.02)	5.56 (0.60)	21.46 (0.47)	**50.16** (0.38)

(b) Autonomous Car Navigation domain

Table 1: Results from evaluating the returns of each method for 2000 episodes. Brackets indicate standard errors. Time for *RA-BAMCP* and *BAMCP* indicates average total computation time per episode to perform simulations. Time for *CVaR VI BAMDP* and *CVaR VI Expected MDP* indicates the time for VI to converge. Time for *CVaR BAMDP PG* indicates the total training time.

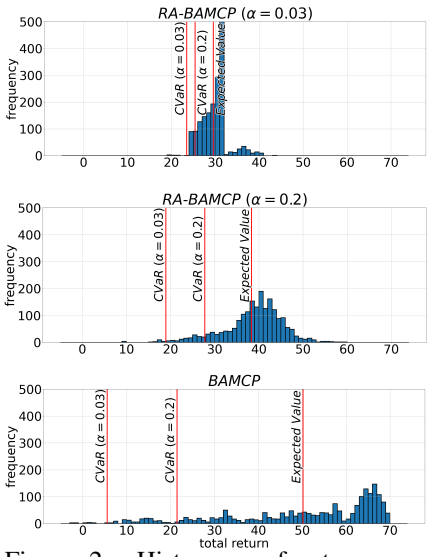


Figure 2: Histogram of returns received in the Autonomous Car Navigation domain. Vertical red lines indicate $\text{CVaR}_{0.03}$, $\text{CVaR}_{0.2}$, and the expected value of the return distributions.

Results in the supplementary material show that our approach using random action expansion rather than Bayesian optimisation attains worse performance for the same computation budget. This demonstrates that Bayesian optimisation for action expansion is critical to our approach.

6 Discussion of Limitations and Conclusion

In this work, we have addressed CVaR optimisation in the BAMDP formulation of model-based Bayesian RL. This is motivated by the aim of mitigating risk due to both epistemic uncertainty over the model, and aleatoric uncertainty due to environment stochasticity. Our experimental evaluation demonstrates that our algorithm, *RA-BAMCP*, outperforms baseline approaches for this problem.

One of the limitations of our work is that our approach cannot use root sampling [16]. Instead, the posterior distribution is computed at each node in the MCTS tree, prohibiting the use of complex priors. In future work, we wish to develop an MCTS algorithm for CVaR optimisation which can use root sampling. Additionally, due to the exponential growth of the BAMDP and the difficulty of optimising CVaR, the scalability of our approach is limited. In future work, we wish to utilise function approximation to improve the scalability of our approach to more complex problems. One direction could be using a neural network to guide search, in the style of AlphaGo [39]. Another direction could be to extend the policy gradient algorithm that we proposed in the experiments section to make use of deep learning.

Acknowledgements and Funding Disclosure

The authors would like to thank Paul Duckworth for valuable feedback on an earlier draft of this work.

This work was supported by a Programme Grant from the Engineering and Physical Sciences Research Council [EP/V000748/1], the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services.

References

[1] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

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

[3] Nicole Bäuerle and Jonathan Ott. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3):361–379, 2011.

[4] Vivek Borkar and Rahul Jain. Risk-constrained Markov decision processes. In *49th IEEE Conference on Decision and Control (CDC)*, pages 2664–2669. IEEE, 2010.

[5] Katherine Chen and Michael Bowling. Tractable objectives for robust policy optimization. In *Advances in Neural Information Processing Systems*, pages 2069–2077, 2012.

[6] Yinlam Chow and Mohammad Ghavamzadeh. Algorithms for CVaR optimization in mdps. In *Advances in neural information processing systems*, pages 3509–3517, 2014.

[7] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

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

[9] William R Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.

[10] Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous upper confidence trees. In *International Conference on Learning and Intelligent Optimization*, pages 433–445. Springer, 2011.

[11] Erick Delage and Shie Mannor. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations research*, 58(1):203–213, 2010.

[12] Michael O Duff. Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes. 2003.

[13] Hannes Eriksson, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis. Sentinel: Taming uncertainty with ensemble-based distributional reinforcement learning. *arXiv preprint arXiv:2102.11075*, 2021.

[14] Sylvain Gelly and David Silver. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[15] Arthur Guez, Nicolas Heess, David Silver, and Peter Dayan. Bayes-adaptive simulation-based search with value function approximation. In *NIPS*, volume 27, pages 451–459, 2014.

[16] Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. *Advances in neural information processing systems*, 25:1025–1033, 2012.

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

[18] Jongmin Lee, Youngsoo Jang, Pascal Poupart, and Kee-Eung Kim. Constrained Bayesian reinforcement learning via approximate linear programming. In *IJCAI*, pages 2088–2095, 2017.

[19] Jongmin Lee, Geon-Hyeong Kim, Pascal Poupart, and Kee-Eung Kim. Monte-carlo tree search for constrained pomdps. In *NeurIPS*, pages 7934–7943, 2018.

[20] Shie Mannor, Duncan Simester, Peng Sun, and John N Tsitsiklis. Bias and variance in value function estimation. In *Proceedings of the twenty-first international conference on Machine learning*, page 72, 2004.

[21] James John Martin. *Bayesian decision problems and Markov chains*. Wiley, 1967.

[22] John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J Kochenderfer. Bayesian optimized Monte Carlo planning. *arXiv preprint arXiv:2010.03597*, 2020.

[23] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(12), 2006.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[25] Philippe Morere, Roman Marchant, and Fabio Ramos. Bayesian optimisation for solving continuous state-action-observation pomdps. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[26] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[27] Takayuki Osogami. Robustness and risk-sensitivity in Markov decision processes. *Advances in Neural Information Processing Systems*, 25:233–241, 2012.

[28] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8522–8528. IEEE, 2019.

[29] Georg Ch Pflug and Alois Pichler. Time-consistent decisions and temporal decomposition of coherent risk functionals. *Mathematics of Operations Research*, 41(2):682–699, 2016.

[30] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2817–2826, 2017.

[31] LA Prashanth. Policy gradients for CVaR-constrained MDPs. In *International Conference on Algorithmic Learning Theory*, pages 155–169. Springer, 2014.

[32] Marc Rigter, Paul Duckworth, Bruno Lacerda, and Nick Hawes. Lexicographic optimisation of conditional value at risk and expected value for risk-averse planning in MDPs. *arXiv preprint arXiv:2110.12746*, 2021.

[33] Marc Rigter, Bruno Lacerda, and Nick Hawes. Minimax regret optimisation for robust planning in uncertain Markov decision processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11930–11938, May 2021.

[34] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

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

[36] Pedro Santana, Sylvie Thiébaux, and Brian Williams. RAO*: An algorithm for chance-constrained POMDP's. 2016.

[37] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2014.

[38] Apoorva Sharma, James Harrison, Matthew Tsao, and Marco Pavone. Robust and adaptive planning under model uncertainty. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 410–418, 2019.

[39] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[40] Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtěch Franc. The SHOGUN machine learning toolbox. *The Journal of Machine Learning Research*, 11:1799–1802, 2010.

[41] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA, 2010. Omnipress.

[42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[44] Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Sequential decision making with coherent risk. *IEEE Transactions on Automatic Control*, 62(7):3323–3338, 2016.

[45] Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2993–2999. AAAI Press, 2015.

[46] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. Worst cases policy gradients. *arXiv preprint arXiv:1911.03618*, 2019.

[47] Eric M Wolff, Ufuk Topcu, and Richard M Murray. Robust control of uncertain Markov decision processes with temporal logic specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3372–3379. IEEE, 2012.

5.1 Limitations and Future Work

There are two key limitations to this work. The first is that it is necessary to specify a suitable prior over the MDP. If a completely uninformative prior is used, this is unlikely to be very meaningful when it comes to optimising risk. It is also difficult to know how to represent the prior such that the posterior can be computed in a scalable way. These issues were not the focus of this piece of research, so we used simple Dirichlet priors for the toy problems we tested on. A more appropriate approach in future work may instead be to use approaches that learn to represent the prior and posterior update over MDPs after repeated interactions with samples of MDPs [178, 108], rather than specifying this a priori.

The second major limitation of this work is that the two-player Monte Carlo tree search algorithm that we proposed is unlikely to scale to larger problems. Indeed, we found that scaling our method beyond the toy problems presented in the paper was difficult. This is because the adversary has a large and continuous action space for perturbing the transition probabilities. Furthermore, because we optimise the static risk objective, the adversary's optimal strategy depends on it's "budget" for perturbing the transition probabilities over the entire trajectory. These two factors make it is very difficult to find the correct policy for the adversary using our approach. If the adversary is suboptimal, then the agent policy does not correctly optimise CVaR in response.

There are two possible approaches to alleviating these scalability issues. The first would be to consider the dynamic perspective of risk, rather than static risk. For dynamic risk, we would not have to keep track of the "budget" of the adversary to perturb the transition probabilities. This is because in the dynamic perspective of risk, the risk measure is applied recursively at each step. This means that we can separately consider an adversarially modified transition distribution at each step, rather than over the entire trajectory. The downside of the dynamic risk approach is that it would be very difficult to interpret what the resulting algorithm achieves, especially in the BAMDP setting.

Another avenue to improving scalability would be to utilise deep RL algorithms for optimising CVaR based on either policy gradients [53] or distributional RL [71]. These algorithms could be combined with methods for BAMDPs where the posterior update is learned via sampling [108]. This approach would stay true to our original aim of jointly mitigating risk due to both the uncertainty over the unknown MDP, and the inherent randomness of MDPs, but would leverage scalable RL methods achieve this goal.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Risk-averse Bayes-adaptive reinforcement learning.
Publication Status	□Published ☑Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Marc Rigter, Bruno Lacerda and Nick Hawes (2021). Risk-averse Bayes-adaptive reinforcement learning. Advances in Neural Information Processing Systems (NeurIPS).

Student Confirmation

Student Name:	Marc Rigter		
Contribution to the Paper	I led the development of the idea in collaboration with Bruno Lacerda and Nick Hawes. I implemented the algorithm and ran the experiments. I led the writing in collaboration with Bruno Lacerda and Nick Hawes.		
Signature		Date	16th November 2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Nick Hawes		
Supervisor comments	I confirm that Marc made a substantial contribution to the publication, and that the description above is accurate.		
Signature		Date	22nd November 2022

This completed form should be included in the thesis, at the end of the relevant chapter.

6

RAMBO-RL: Robust Adversarial Model-Based Offline Reinforcement Learning

The two previous pieces of work assumed that prior information about the MDP was specified either in the form of a set of plausible MDPs, or a suitable prior over MDPs. In the following two chapters, we address the offline reinforcement learning setting, where the uncertainty over the MDP is specified by a fixed dataset of experience in the environment. This source of knowledge about the environment is indicated in Table 1.1. This is arguably a more pragmatic problem formulation, as most applications of sequential decision-making algorithms in the real world will need to be based on data collected from the environment.

Offline RL avoids the need for the extensive online exploration required by standard online RL methods by leveraging pre-existing datasets. We consider

Table 1.1: Summary of the attributes of each chapter.

	Chapter 3	**Chapter 4**	**Chapter 5**	**Chapter 6**	**Chapter 7**	**Chapter 8**
Prior knowledge	MDP fully known	MDP uncertainty set	Prior over MDPs	Fixed dataset	Fixed dataset	Expert demonstrator
Considers aleatoric uncertainty	Yes	No	Yes	No	Yes	No
Considers epistemic uncertainty	No	Yes	Yes	Yes	Yes	Yes
Risk or robustness	Risk	Robustness	Risk	Robustness	Risk	N/A
Paradigm	Tabular	Tabular	Tabular	Function Approx.	Function Approx.	Function Approx.

model-based approaches, which learn a model of the environment dynamics using the dataset, and use that model to generate synthetic data to train a policy.

As discussed in Section 2.5, the issue with directly applying this approach is that the policy may learn to choose actions that exploit errors in the dynamics model. Instead, we would like to ensure that the policy takes actions that are likely to perform well in the real environment, as evidenced by the dataset. The contribution of this work is to propose a new approach to address the issue of model exploitation without the need for explicit uncertainty estimation.

We address an existing formulation of model-based offline RL that poses the problem as solving a type of robust MDP, where the goal is the find the policy with the best expected value in the worst-case instantiation of the MDP. The uncertainty set for the robust MDP is defined such that it includes all models that have a low prediction error with respect to the dataset. This problem formulation has been analysed theoretically in prior work [179]. As indicated by Table 1.1, this chapter therefore proposes a robust approach to addressing epistemic uncertainty. Because the expected value is optimised within the worst-case MDP, this approach does not take into consideration aleatoric uncertainty.

For the remainder of this thesis, we no longer focus on tabular MDPs. Instead, we develop methods that employ deep RL to utilise powerful neural network function approximators and achieve scalability to problems with large or continuous state spaces. The use of function approximation in the remaining chapters of this thesis is indicated by the "Paradigm" row of Table 1.1.

To arrive at a scalable solution to the robust MDP formulation that we address, we propose an approach inspired by Robust Adversarial RL [98]. We alternate between optimising the policy to increase the expected reward in the model, and optimising the model to decrease it. However, the model is constrained such that it must still accurately predict the transitions within the dataset. Training the model in this way means that it predicts accurate transitions for state-action pairs covered by the dataset, but generates pessimistic synthetic transitions for areas far from the

dataset. This prevents the policy from exploiting errors in the model, because in regions where there is little or no data, the model generates pessimistic transitions.

We evaluate our approach on standard benchmarks for offline RL, and compare our algorithm to a number of existing methods. The results demonstrate that our approach achieves state-of-the-art performance on these benchmarks. We also analyse the characteristics of our approach on a toy example. We find that initially, the value function is optimistic and pessimism is introduced gradually as the model is trained adversarially. Our results suggest that this initial optimism makes it less likely that the policy becomes stuck in a poor local minimum, compared to the most similar existing approach.

Marc Rigter, Bruno Lacerda and Nick Hawes (2022). RAMBO-RL: Robust adversarial model-based offline reinforcement learning. *Advances in Neural Information Processing Systems* (NeurIPS).

RAMBO-RL: Robust Adversarial Model-Based Offline Reinforcement Learning

Marc Rigter, Bruno Lacerda, Nick Hawes
Oxford Robotics Institute
University of Oxford
{mrigter, bruno, nickh}@robots.ox.ac.uk

Abstract

Offline reinforcement learning (RL) aims to find performant policies from logged data without further environment interaction. Model-based algorithms, which learn a model of the environment from the dataset and perform conservative policy optimisation within that model, have emerged as a promising approach to this problem. In this work, we present Robust Adversarial Model-Based Offline RL (RAMBO), a novel approach to model-based offline RL. We formulate the problem as a two-player zero sum game against an adversarial environment model. The model is trained to minimise the value function while still accurately predicting the transitions in the dataset, forcing the policy to act conservatively in areas not covered by the dataset. To approximately solve the two-player game, we alternate between optimising the policy and adversarially optimising the model. The problem formulation that we address is theoretically grounded, resulting in a probably approximately correct (PAC) performance guarantee and a pessimistic value function which lower bounds the value function in the true environment. We evaluate our approach on widely studied offline RL benchmarks, and demonstrate that it outperforms existing state-of-the-art baselines.

1 Introduction

Reinforcement learning (RL) [61] has achieved state-of-the-art performance on many sequential decision-making problems [40, 45, 59]. However, the need for extensive exploration prohibits the application of RL to many real world domains where such exploration is costly or dangerous. Offline RL [33, 35] overcomes this limitation by learning policies from static, pre-recorded datasets.

Online RL algorithms perform poorly in the offline setting due to the distributional shift between the state-action pairs in the dataset and those taken by the learnt policy. Thus, an important aspect of offline RL is to introduce conservatism to prevent the learnt policy from executing state-action pairs which are out of distribution. Model-free offline RL algorithms [15, 23, 27, 29, 31, 72] train a policy from only the data present in the fixed dataset, and incorporate conservatism either into the value function or by directly constraining the policy.

On the other hand, model-based offline RL algorithms [76, 75, 25, 63, 39] use the dataset to learn a model of the environment, and train a policy using additional synthetic data generated from that model. By training on additional synthetic data, model-based algorithms can potentially generalise better to states not present in the dataset, or to solving new tasks. Previous approaches to model-based offline RL incorporate conservatism by estimating the uncertainty in the model and applying reward penalties for state-action pairs that have high uncertainty [76, 25]. However, uncertainty estimation can be unreliable for neural network models [75, 37]. Like recent work [75], we propose an approach for offline model-based RL which *does not require uncertainty estimation*.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

In this work we present Robust Adversarial Model-Based Offline (RAMBO) RL, a new algorithm for model-based offline RL. RAMBO incorporates conservatism by modifying the *transition dynamics* of the learnt Markov decision process (MDP) model in an adversarial manner. We formulate the problem of offline RL as a zero-sum game against an adversarial environment. To solve the resulting maximin optimisation problem, we alternate between optimising the agent and optimising the adversary in the style of Robust Adversarial RL (RARL) [50]. Unlike existing RARL approaches, our model-based approach forgoes the need to define and train an adversary policy, and instead only learns an adversarial model of the MDP. We train the agent policy with an actor-critic algorithm using synthetic data generated from the model in addition to data sampled from the dataset, similar to Dyna [60] and a number of recent methods [22, 76, 25, 75]. We update the environment model so that it reduces the value function for the agent policy, while still accurately predicting the transitions in the dataset. As a result, our approach introduces conservatism by generating *pessimistic synthetic transitions* for state-action pairs which are out-of-distribution. The theoretical formulation of offline RL that our algorithm addresses yields a PAC bound for the performance gap with respect to any policy covered by the dataset, and a pessimistic value function that lower bounds the value function in the true environment.

In summary, the main contributions of this work are:

- RAMBO, a novel and theoretically-grounded model-based offline RL algorithm which enforces conservatism by training an adversarial dynamics model.
- Adapting the Robust Adversarial RL approach to model-based offline RL by proposing a new formulation of RARL, where instead of defining and training an adversary policy, we directly train the model adversarially.

In our experiments we demonstrate that RAMBO outperforms current state-of-the-art algorithms on the D4RL benchmarks [14]. Furthermore, we provide ablation results which show that training the model adversarially is crucial to the strong performance of RAMBO.

2 Related Work

Offline RL: Offline RL addresses the problem of learning policies from fixed datasets, and has been applied to domains such as healthcare [42, 57], natural language processing [24, 23], and robotics [30, 38, 51]. Model-free offline RL algorithms do not require a learnt model. Approaches for model-free offline RL include importance sampling algorithms [36, 41], constraining the learnt policy to be similar to the behaviour policy [16, 28, 72, 23, 58, 15], incorporating conservatism into the value function during training [9, 27, 31, 73], using uncertainty quantification to generate more robust value estimates [1, 2, 29], or applying only a single iteration of policy iteration [7, 49]. In contrast, model-based approaches learn a model of the environment and generate synthetic data from that model [60] to optimise a policy using either planning [4] or RL algorithms [25, 76, 75]. By training a policy on additional synthetic data, model-based approaches have the potential for broader generalisation and for solving new tasks [6, 76]. A simple approach to ensuring conservatism is to constrain the policy to be similar to the behaviour policy in the same fashion as some model-free approaches [8, 39, 63]. Another approach is to apply reward penalties for executing state-action pairs with high uncertainty in the environment model [25, 74, 76]. However, this requires explicit uncertainty estimates which may be unreliable for neural network models [17, 37, 46, 75]. COMBO [75] obviates the need for uncertainty estimation in model-based offline RL by adapting model-free techniques [31] to regularise the value function for out-of-distribution samples. Like COMBO, our approach does not require uncertainty estimation.

Most approaches to model-based offline RL use maximum likelihood estimates (MLE) of the MDP trained using standard supervised learning [4, 39, 63, 76, 75]. However, other methods have been proposed to learn models which are more suitable for offline policy optimisation. One approach is to reweight the loss function to ensure the model is accurate under the state-action distribution generated by the policy [34, 53, 20]. In contrast, our approach produces pessimistic synthetic transitions when out-of-distribution.

Most related to our work is a recent paper [66] which introduces the maximin formulation of offline RL that we address. This existing work motivates our approach theoretically by showing that the problem formulation obtains probably approximately correct (PAC) guarantees. However, [66]

only addresses the theoretical aspects of the problem formulation and does not propose a practical algorithm. In this work, we propose a practical RL algorithm to solve the maximin formulation of model-based offline RL.

Robust RL: Algorithms for Robust MDPs [5, 12, 21, 43, 54, 64, 70] find the policy with the best worst-case performance over a set of possible MDPs. Typically, it is assumed that the uncertainty set of MDPs is specified a priori. To eliminate the need to specify the set of possible MDPs, model-free approaches to Robust MDPs [68, 55] instead assume that samples can be drawn from a misspecified MDP which is similar to the true MDP. As our work addresses offline RL, we assume that we have a fixed dataset from the true MDP.

Our approach is conceptually similar to Robust Adversarial RL (RARL) [50], a method proposed to improve the robustness of RL policies in the *online* setting. RARL is posed as a two-player zero-sum game where the agent plays against an adversary which perturbs the environment. Formulations of model-free RARL differ in how they define the action space of the adversary. Options include allowing the adversary to apply perturbation forces to the simulator [50], add noise to the agent's actions [65], or periodically take over control [48]. A model-based approach to RARL is proposed in [13], which learns an optimistic and pessimistic model to encourage online exploration. However, this existing approach requires uncertainty estimation as well as an adversarial policy to be learnt in *addition to the model*. Our work follows the paradigm of RARL and alternates between agent and adversarial updates in a maximin formulation. We adapt model-based RARL to the offline setting and propose an alternative formulation: we eliminate the need to learn an adversary policy and instead *directly modify the MDP model adversarially*.

3 Preliminaries

MDPs and Offline RL: An MDP is defined by the tuple, $M = (S, A, T, R, \mu_0, \gamma)$. S and A denote the state and action spaces respectively, $R(s, a)$ is the reward function, $T(s'|s, a)$ is the transition function, μ_0 is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor. In this work we consider Markovian policies, $\pi \in \Pi$, which map from each state to a distribution over actions. We denote the (improper) discounted state visitation distribution of a policy by $d^\pi_M(s) := \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s|\pi, M)$, where $\Pr(s_t = s|\pi, M)$ is the probability of reaching state s at time t by executing policy π in M. The improper state-action visitation distribution is $d^\pi_M(s, a) = \pi(a|s) \cdot d^\pi_M(s)$. We also denote the normalised state-action visitation distribution by $\tilde{d}^\pi_M(s, a) = (1 - \gamma) \cdot d^\pi_M(s, a)$.

The value function, $V^\pi_M(s)$, represents the expected discounted return from executing π from state s in M: $V^\pi_M(s) = \mathbb{E}_{\pi,M}\big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\big]$. We write V^π_M to indicate the value function under the initial state distribution, i.e. $V^\pi_M = \sum_{s\in S} \mu_0(s) V^\pi_M(s)$. The standard objective for MDPs is to find the policy which maximises V^π_M. The state-action value function, $Q^\pi_M(s, a)$, is the expected discounted cumulative reward from taking action a at state s and then executing π thereafter.

In offline RL we only have access to a fixed dataset of transitions from the MDP, $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{|\mathcal{D}|}$. The goal of offline RL is to find the best possible policy using the fixed dataset.

Model-Based Offline RL Algorithms: Model-based approaches to offline RL use a model of the MDP to help train a policy. The dataset is used to learn a dynamics model, $\widehat{T}$, which is typically trained via maximum likelihood estimation: $\min_{\widehat{T}} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\big[-\log \widehat{T}(s'|s, a)\big]$. A model of the reward function, $\widehat{R}(s, a)$, can also be learnt if it is unknown. The estimated MDP, $\widehat{M} = (S, A, \widehat{T}, \widehat{R}, \mu_0, \gamma)$, has the same state and action space as the true MDP but uses the learnt transition and reward functions. Thereafter, any planning or RL algorithm can be used to recover optimal policy in the learnt model, $\widehat{\pi} = \arg\max_{\pi\in\Pi} V^\pi_{\widehat{M}}$.

Unfortunately, directly applying this approach to the offline RL setting does not perform well due to distributional shift. In particular, if the dataset does not cover the entire state-action space, the model will inevitably be inaccurate for some state-action pairs. Thus, naive policy optimisation on a learnt model in the offline setting can result in *model exploitation* [22, 32, 53]. To mitigate this issue, we propose the novel approach of enforcing conservatism by adversarially modifying the transition dynamics of $\widehat{M}$.

In line with existing works [76, 75, 8], we use model-based policy optimisation (MBPO) [22] to learn the optimal policy for $\widehat{M}$. MBPO utilises a standard actor-critic RL algorithm. However, the value function is trained using an augmented dataset $\mathcal{D} \cup \mathcal{D}_{\widehat{M}}$, where $\mathcal{D}_{\widehat{M}}$ is synthetic data generated by simulating rollouts in the learnt model. To generate the synthetic data, MBPO performs k-step rollouts in $\widehat{M}$ starting from states $s \in \mathcal{D}$, and adds this data to $\mathcal{D}_{\widehat{M}}$. To train the policy, minibatches of data are drawn from $\mathcal{D} \cup \mathcal{D}_{\widehat{M}}$, where each datapoint is sampled from the real data, $\mathcal{D}$, with probability f, and from $\mathcal{D}_{\widehat{M}}$ with probability $1 - f$.

Robust Adversarial Reinforcement Learning: RARL addresses the problem of finding a robust agent policy, π, in the online RL setting by posing the problem as a two-player zero sum game against adversary policy, $\bar{\pi}$:

$$\pi = \arg\max_{\pi \in \Pi} \min_{\bar{\pi} \in \bar{\Pi}} V_M^{\pi, \bar{\pi}} \tag{1}$$

where $V_M^{\pi,\bar{\pi}}$ is the expected value from executing π and $\bar{\pi}$ in environment M. Different approaches define the action space for $\bar{\pi}$ in different ways, as discussed in Section 2. For a scalable approximation to the optimisation problem in Equation 1, algorithms for RARL alternate between applying steps of stochastic gradient ascent to the agent's policy to increase the expected value, and stochastic gradient descent to the adversary's policy to decrease the expected value. In our work, we follow the RARL paradigm of alternating between agent and adversarial updates and adapt it to the model-based offline setting. Instead of defining a separate adversary policy, we treat the *model itself* as the policy to be adversarially trained.

4 Problem Formulation

For the sake of generality, we assume that both the transition function and reward function are unknown. Hereafter, we write $\widehat{T}$ to denote both the learnt dynamics and reward function, where $\widehat{T}(s', r|s, a)$ is the probability of receiving reward r and transitioning to s' after executing (s, a). We address the maximin formulation of offline RL recently proposed by [66]:

Problem 1. *For some dataset, $\mathcal{D}$, and some fixed constant $\xi > 0$, find the policy π defined by*

$$\pi = \arg\max_{\pi \in \Pi} \min_{\widehat{T} \in \mathcal{M}_{\mathcal{D}}} V_{\widehat{T}}^{\pi}, \text{ where} \tag{2}$$

$$\mathcal{M}_{\mathcal{D}} = \Big\{ \widehat{T} \mid \mathbb{E}_{\mathcal{D}}\big[\mathrm{TV}(\widehat{T}_{\mathrm{MLE}}(\cdot|s,a), \widehat{T}(\cdot|s,a))^2\big] \leq \xi \Big\}, \tag{3}$$

where $\mathrm{TV}(P_1, P_2)$ is the total variation distance between distributions P_1 and P_2, and $\widehat{T}_{\mathrm{MLE}}$ denotes the maximum likelihood estimate of the MDP given the offline dataset, $\mathcal{D}$.

Thus, the set defined in Equation 3 contains MDPs which are similar to the maximum likelihood estimate under state-action pairs in $\mathcal{D}$. However, because the expectation in Equation 3 is taken under $\mathcal{D}$, there is no restriction on $\widehat{T}$ for regions of the state-action space not covered by $\mathcal{D}$. We present a brief overview of the theoretical guarantees from [66] in the following subsection.

Remark 1. Note that Problem 1 differs from the pessimistic MDP formulations introduced by MOPO [76] and MOReL [25]. Problem 1 considers the worst-case transition dynamics, while the pessimistic MDPs constructed by MOPO and MOReL only modify the reward function by applying reward penalties for state-action pairs with high uncertainty.

4.1 Theoretical Motivation

The theoretical analysis from [66] shows that solving Problem 1 outputs a policy that with high probability is approximately as good as any policy with a state-action distribution that is covered by the dataset. This is formally stated in the following theorem.

Theorem 1 (PAC guarantee from [66], Theorem 5)**.** *Denote the true MDP transition function by T, and let $\mathcal{M}$ denote a hypothesis class of MDP models such that $T \in \mathcal{M}$. Let π denote the solution to Problem 1 for dataset $\mathcal{D}$. Then with probability $1 - \delta$ for any policy, $\pi^* \in \Pi$, we have*

$$V_T^{\pi^*} - V_T^{\pi} \leq (1-\gamma)^{-2} c_1 \sqrt{C_{\pi^*}} \sqrt{G_{\mathcal{M}_1} + G_{\mathcal{M}_2} + \xi_n^2 + \frac{\ln(c/\delta)}{|\mathcal{D}|}}, \text{ where}$$

$$C_{\pi^*} = \max_{T' \in \mathcal{M}} \frac{\mathbb{E}_{(s,a)\sim\tilde{d}^{\pi^*}_{T}}\left[\text{TV}(T'(\cdot|s,a), T(\cdot|s,a))^2\right]}{\mathbb{E}_{(s,a)\sim\rho}\left[\text{TV}(T'(\cdot|s,a), T(\cdot|s,a))^2\right]},$$

where ρ is the state-action distribution from which $\mathcal{D}$ was sampled, and c and c_1 are universal constants. We refer the reader to Appendix A of [66] for the definitions of $G_{\mathcal{M}_1}$, $G_{\mathcal{M}_2}$, and ξ_n.

The quantity C_{π^*} is upper bounded by the maximum density ratio between the comparator policy, π^*, and the offline distribution, i.e. $C_{\pi^*} \leq \max_{(s,a)} \tilde{d}^{\pi^*}_{T}(s,a)/\rho(s,a)$. It represents the discrepancy between the distribution of data in the dataset compared to the visitation distribution of policy π^*. Theorem 1 shows that if we find a policy by solving Problem 1, the performance gap of that policy is bounded with respect to any other policy π^* that has a state-action distribution which is covered by the dataset.

Furthermore, the value function under the worst-case model in the set defined by Problem 1 is a lower bound on the value function in the true environment, as stated by Proposition 1.

Proposition 1 (Pessimistic value function)**.** *Let T denote the true transition function for some MDP, and let $\mathcal{M}_{\mathcal{D}}$ be the set of MDP models defined in Equation 3. Then for any policy π, with probability $1-\delta$ we have that*

$$\min_{\widehat{T} \in \mathcal{M}_{\mathcal{D}}} V^{\pi}_{\widehat{T}} \leq V^{\pi}_{T}.$$

Proposition 1 follows from the fact that $T \in \mathcal{M}_{\mathcal{D}}$ with high probability, which is proven in [66] (Appendix E.2). Proposition 1 shows that we can expect the performance of any policy in the true MDP to be at least as good as the value in the worst-case model defined in Problem 1.

While [66] provides the theoretical motivation for solving Problem 1, it does not propose a practical algorithm. In this work, we focus on developing a practical approach to solving Problem 1.

5 RAMBO-RL

In this section, we present Robust Adversarial Model-Based Offline RL (RAMBO), our algorithm for solving Problem 1. The main difficulty with solving Problem 1 is that it is unclear how to find the worst-case MDP in the set defined in Equation 3. To arrive at a scalable solution, we propose a novel approach which is in the spirit of RARL. We alternate between optimising the agent policy to increase the expected value, and adversarially optimising the model to decrease the expected value. In this section, we first describe how we compute the gradient to adversarially train the model. Then, we discuss how to ensure that the model remains approximately within the constraint set defined in Problem 1. Finally, we present our overall algorithm.

5.1 Model Gradient

We propose a policy gradient-inspired approach to adversarially optimise the model. Typically, policy gradient algorithms are used to modify the distribution over actions taken by a policy at each state [62, 71]. In contrast, the update that we propose modifies the likelihood of the successor states and rewards in the MDP model.

We assume that the MDP model is defined by parameters, ϕ, and we write $\widehat{T}_\phi$ to indicate this. We denote by V^{π}_{ϕ} the value function for policy π in model $\widehat{T}_\phi$. To approximately find $\min_{\widehat{T}_\phi \in \mathcal{M}_{\mathcal{D}}} V^{\pi}_{\phi}$ as required by Problem 1 via gradient descent, we wish to compute the gradient of the model parameters that reduces the value of the policy within the model, i.e. $\nabla_\phi V^{\pi}_{\phi}$.

Proposition 2 (Model Gradient)**.** *Let ϕ denote the parameters of a parametric MDP model $\widehat{T}_\phi$, and let V^{π}_{ϕ} denote the value function for policy π in $\widehat{T}_\phi$. Then:*

$$\nabla_\phi V^{\pi}_{\phi} = \mathbb{E}_{s\sim d^{\pi}_{\phi}, a\sim\pi, (s',r)\sim\widehat{T}_\phi}\left[(r + \gamma V^{\pi}_{\phi}(s')) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a)\right] \tag{4}$$

The proof of Proposition 2 is given in Appendix A. We can subtract the baseline $Q^{\pi}_{\phi}(s,a)$ without biasing the gradient estimate (see Appendix A.1 for details):

$$\nabla_\phi V^{\pi}_{\phi} = \mathbb{E}_{s\sim d^{\pi}_{\phi}, a\sim\pi, (s',r)\sim\widehat{T}_\phi}\left[(r + \gamma V^{\pi}_{\phi}(s') - Q^{\pi}_{\phi}(s,a)) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a)\right] \tag{5}$$

The Model Gradient differs from the standard policy gradient in that a) it is used to update the likelihood of successor states in the model, rather than actions in a policy, and b) the "advantage" term $r + \gamma V^\pi_\phi(s') - Q^\pi_\phi(s,a)$ compares the utility of receiving reward r and transitioning to s' to the expected value of state action pair (s,a). In contrast, the standard advantage term, $Q(s,a) - V(s)$ [56], compares the value of executing state-action pair (s,a) to the value at s. To estimate V^π_ϕ and Q^π_ϕ, we use the critic learnt by the actor-critic algorithm used for policy optimisation. Thus, we use the critic *both* for training the policy and adversarially training the model.

Remark 2. The Model Gradient can be thought of as a specific instantiation of the policy gradient if we view $\widehat{T}_\phi$ as a adversarial "policy" on an augmented MDP, M^+, i.e. $\widehat{T}_\phi : S^+ \to \text{Dist}(A^+)$. The augmented state space, $S^+ = S \times A$ includes the state in the original MDP augmented by the action taken by the agent. The augmented action space consists of the reward applied and the successor state, $A^+ = S \times [R_{\text{min}}, R_{\text{max}}]$. Thus, we can think of this approach as an instantiation of RARL in which the adversary policy ($\bar{\pi}$ in Equation 1) that we train is the *model itself*.

5.2 Adversarial Model Training

If we were to update the model using Equation 5 alone, this would allow the model to be modified arbitrarily such that the value function in the model is reduced. However, the set of plausible MDPs given by Equation 3 states that over the dataset, $\mathcal{D}$, the model $\widehat{T}_\phi$ should be close to the maximum likelihood estimate, $\widehat{T}_{\text{MLE}}$. Specifically, in the inner optimisation of Problem 1 we wish to find a solution to the constrained optimisation problem

$$\min_{\widehat{T}_\phi} V^\pi_\phi, \quad s.t.\ \mathbb{E}_{\mathcal{D}}\big[\text{TV}(\widehat{T}_{\text{MLE}}(\cdot|s,a), \widehat{T}_\phi(\cdot|s,a))^2\big] \leq \xi. \tag{6}$$

The Lagrangian relaxation leads to the unconstrained problem

$$\max_{\lambda \geq 0} \min_{\widehat{T}_\phi} \Big(L(\hat{T}, \lambda) := V^\pi_\phi + \lambda\big(\mathbb{E}_{\mathcal{D}}\big[\text{TV}(\widehat{T}_{\text{MLE}}(\cdot|s,a), \widehat{T}_\phi(\cdot|s,a))^2\big] - \xi\big)\Big), \tag{7}$$

where λ is the Lagrange multiplier. Rather than optimising the Lagrange multiplier, we find that in practice fixing λ to apply a constant weighting between the two terms works well with minimal tuning. To facilitate easier tuning of the learning rate, in our implementation we apply the weighting constant to the value function term rather than the model term, which is equivalent up to a scaling factor. This leads to

$$\min_{\widehat{T}_\phi} \Big(\lambda V^\pi_\phi + \mathbb{E}_{\mathcal{D}}\big[\text{TV}(\widehat{T}_{\text{MLE}}(\cdot|s,a), \widehat{T}_\phi(\cdot|s,a))^2\big]\Big). \tag{8}$$

We aim to make our algorithm efficient and simple to implement. Therefore, rather than minimising the TV distance between the model and MLE model as prescribed by Equation 8, we directly optimise the standard MLE loss. This leads to the final loss function:

$$\mathcal{L}_\phi = \lambda V^\pi_\phi - \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\big[\log \widehat{T}_\phi(s', r|s,a)\big]. \tag{9}$$

Thus, the loss function for the model in Equation 9 simply adds the adversarial term to the standard MLE loss. By minimising the loss function in Equation 9, the model is trained to a) predict the transitions within the dataset, and b) reduce the value function of the policy, with λ determining the tradeoff between these two objectives. Choosing λ to be small ensures that the MLE term dominates for transitions within $\mathcal{D}$, ensuring that the model fits the dataset accurately. Because the MLE term is only computed over $\mathcal{D}$, the adversarial term dominates outside of the dataset meaning that the model is modified adversarially for transitions outside of the dataset.

To estimate the gradient of the loss function in Equation 9 for stochastic gradient descent, we sample a minibatch of transitions from $\mathcal{D}$ to estimate the MLE term. The gradient for the value function term is computed using the Model Gradient. The transitions used to compute the Model Gradient term must be sampled under the current policy and model (Equation 5). Therefore, to estimate the Model Gradient term, we generate a minibatch of transitions by simulating the current policy in $\widehat{T}_\phi$.

Like previous works [10, 76, 75, 8], we represent the dynamics model using an ensemble of neural networks. Each neural network produces a Gaussian distribution over the next state and reward:

$\widehat{T}_\phi(s', r|s, a) = \mathcal{N}(\mu_\phi(s, a), \Sigma_\phi(s, a))$. A visualisation of the result of adversarially training the dynamics model can be found in Appendix C.3.

Normalisation The composite loss function in Equation 9 comprises two terms which may have different magnitudes across domains depending on the scale of the states and rewards. To enable easier tuning of the adversarial loss weighting, λ, across different domains we perform the following normalisation procedure. Prior to training, we normalise the states in $\mathcal{D}$ in the manner proposed in [15], by subtracting the mean and dividing by the standard deviation of each state dimension in the dataset. Additionally, when computing the gradient in Equation 5 we normalise the advantage terms, $r + \gamma V^\pi_\phi(s') - Q^\pi_\phi(s, a)$, according to the mean and standard deviation across each minibatch. Advantage normalisation is common practice in policy gradient RL implementations [3, 52].

5.3 Algorithm

We are now ready to present our overall approach in Algorithm 1. The first step of RAMBO is to pretrain the environment dynamics model using standard MLE (Line 1). Thereafter, the algorithm follows the format of RARL. At each iteration, we apply gradient updates to the agent to increase the expected value, followed by gradient updates to the model to decrease the expected value.

Prior to each agent update, we generate synthetic k-step rollouts starting from states in $\mathcal{D}$ by simulating rollouts in the current MDP model $\widehat{T}_\phi$. This data is added to the synthetic dataset $\mathcal{D}_{\widehat{T}_\phi}$ (Line 3). Following previous approaches [76, 75, 22] we only store data from recent iterations in $\mathcal{D}_{\widehat{T}_\phi}$. The agent's policy and value functions are trained with an off-policy actor-critic algorithm using samples from $\mathcal{D} \cup \mathcal{D}_{\widehat{T}_\phi}$ (Line 4). In our implementation, we use soft actor-critic (SAC) [19] for agent training. To update the model to minimise the loss in Equation 9 (Line 5), we sample data from $\mathcal{D}$ to estimate the gradient for the MLE component. To compute the adversarial component we generate samples by simulating the current policy and model, and utilise the value function learnt by the agent to compute the gradient according to Equation 5.

Algorithm 1 RAMBO-RL

Require: Normalised dataset, $\mathcal{D}$;

1: $\widehat{T}_\phi \leftarrow$ MLE dynamics model.
2: **for** $i = 1, 2, \ldots, n_{\text{iter}}$ **do**
3: Generate synthetic k-step rollouts. Add transition data to $\mathcal{D}_{\widehat{T}_\phi}$.
4: *Agent update*: Update π and Q^π_ϕ with an actor critic algorithm, using samples from $\mathcal{D} \cup \mathcal{D}_{\widehat{T}_\phi}$.
5: *Adversarial model update*: Update $\widehat{T}_\phi$ according to Eq. 9, using samples from $\mathcal{D}$ for the MLE component, and the current critic Q^π_ϕ and synthetic data sampled from π and $\widehat{T}_\phi$ for the adversarial component.

6 Experiments

In our experiments, we aim to: a) evaluate how well RAMBO performs compared to state-of-the-art baselines, b) examine whether RAMBO can be tuned offline, c) determine the impact of adversarial training on the performance of the algorithm, and d) investigate the difference between RAMBO and COMBO, the most similar prior algorithm. The code for our experiments is available at github.com/marc-rigter/rambo. We evaluate our approach on the following domains.

MuJoCo There are three different environments representing different robots (*HalfCheetah, Hopper, Walker2D*), each with 4 datasets (*Random, Medium, Medium-Replay, Medium-Expert*). *Random* contains transitions collected by a random policy. *Medium* contains transitions collected by an early-stopped SAC policy. *Medium-Replay* consists of the replay buffer generated while training the *Medium* policy. The *Medium-Expert* dataset contains a mixture of suboptimal and expert data.

AntMaze The agent controls a robot and navigates to reach a goal, receiving a sparse reward only if the goal is reached. There are three different layouts of maze (*Umaze, Medium, Large*), and different dataset types (*Fixed, Play, Diverse*) which differ in terms of the variety of start and goal locations used to collect the dataset. The MuJoCo and AntMaze benchmarks are from D4RL [14].

Hyperparameter Details The base hyperparameters that we use for RAMBO mostly follow those used in SAC [19] and COMBO [75]. We find that the performance of RAMBO is sensitive to the choice of rollout length, k, consistent with findings in previous works [22, 37]. The other critical parameter for RAMBO is the choice of the adversarial weighting, λ.

For each dataset, we choose the rollout length and the adversarial weighting from one of three possible configurations: $(k, \lambda) \in \{(2, \text{3e-4}), (5, \text{3e-4}), (5, 0)\}$. We included $(k, \lambda) = (5, 0)$ as we found that an adversarial weighting of 0 worked well for some datasets. For the MuJoCo datasets we performed model rollouts using the current policy, and initialised the policy using behaviour cloning (BC) which is a common practice in offline RL [25, 74]. Ablation results in Appendix C.4 indicate that the BC initialisation results in a small improvement. For the AntMaze datasets we used a random rollout policy and a randomly initialised policy as we found that this performed better. Further details about the hyperparameters are in Appendix B.

Evaluation We present two different evaluations of our approach: RAMBO and RAMBO$^{\text{OFF}}$. For RAMBO, we ran each of the three hyperparameter configurations for five seeds each, and report the best performance across the three configurations. Thus, our evaluation of RAMBO utilises limited online tuning which is the most common practice among existing model-based offline RL algorithms [25, 37, 39, 76]. The performance obtained for each of the hyperparameter configurations is included in Appendix C.2.

Offline hyperparameter selection is an important topic in offline RL [47, 77]. Therefore, we present additional results for RAMBO$^{\text{OFF}}$ where we select between the three choices of hyperparameters offline using a simple heuristic (details in Appendix B.5) based on the magnitude and stability of the Q-values during offline training. We first select the hyperparameters using the heuristic, and then rerun each dataset for 5 seeds to generate the results for RAMBO$^{\text{OFF}}$.

Baselines We compare RAMBO against state-of-the-art model-based (COMBO [75], RepB-SDE [34], MOReL [25], and MOPO [76]) and model-free (CQL [31], IQL [28], and TD3+BC [15]) offline RL algorithms. We provide results for all algorithms for the MuJoCo-v2 D4RL datasets and the AntMaze-v0 datasets (details in Appendix B.7).

6.1 Results

D4RL Performance The results in Table 1 show that RAMBO achieves the best total score for the MuJoCo locomotion domains, outperforming existing state-of-the-art methods. Furthermore, RAMBO obtains the best overall score on both the Medium and Medium-Replay dataset types. For the random datasets, RAMBO is outperformed only by MOReL. This shows that RAMBO performs very well for datasets that are either noisy or consist of suboptimal data.

For the Medium-Expert datasets, RAMBO is outperformed by most of the baseline algorithms, suggesting that RAMBO is less suitable for high-quality datasets. However, for the Medium-Expert datasets, simpler approaches such as performing behaviour cloning on the best 10% of trajectories can be used to achieve stronger performance than offline RL methods [28]. Therefore, the suboptimal performance of RAMBO on the Medium-Expert datasets is less of a concern, as applying offline RL algorithms may not be the most suitable approach for these high-quality datasets.

For AntMaze, the model-based algorithms perform considerably less well than the model-free approaches. Unlike the other model-based approaches, RAMBO at least scores greater than zero for most of the datasets. Our results echo previous findings that model-based approaches struggle to perform well in the AntMaze domains, potentially because model-based algorithms are too aggressive and collide with walls [67]. Recent work [67] showed that *reverse* rollouts can lead to stronger performance for model-based methods in these domains. In future work, we wish to investigate whether combining RAMBO with reverse rollouts improves the performance for AntMaze.

Offline Tuning In Table 1, we also present results for RAMBO$^{\text{OFF}}$ where the final hyperparameters are chosen using the heuristic described in Appendix B.5. We see that there is a slight degradation in the performance relative to RAMBO, which uses online tuning. However, RAMBO$^{\text{OFF}}$ still achieves comparable performance to the best existing approaches on the MuJoCo datasets. This suggests that suitable hyperparameters for RAMBO can reliably be chosen using our offline heuristic.

Table 1: Results for the D4RL benchmark using the normalisation procedure proposed by [14]. We report the normalised performance during the last 10 iterations of training averaged over 5 seeds. $\pm$ captures the standard deviation over seeds. Highlighted numbers indicate results within 2% of the most performant algorithm. * indicates the total without random datasets.

		Ours		Model-based baselines				Model-free baselines			
		RAMBO	**RAMBO$^{\text{OFF}}$**	**RepB-SDE**	**COMBO**	**MOPO**	**MOReL**	**CQL**	**IQL**	**TD3+BC**	**BC**
Random	HalfCheetah	40.0 ± 2.3	33.5 ± 2.6	32.9	38.8	35.4	25.6	19.6	-	11.0	2.1
	Hopper	21.6 ± 8.0	15.5 ± 9.4	8.6	17.9	4.1	53.6	6.7	-	8.5	9.8
	Walker2D	11.5 ± 10.5	0.2 ± 0.6	21.1	7.0	4.2	37.3	2.4	-	1.6	1.6
Medium	HalfCheetah	77.6 ± 1.5	71.0 ± 3.0	49.1	54.2	69.5	42.1	49.0	47.4	48.3	36.1
	Hopper	92.8 ± 6.0	91.2 ± 16.3	34.0	94.9	48.0	95.4	66.6	66.3	59.3	29.0
	Walker2D	86.9 ± 2.7	89.1 ± 2.7	72.1	75.5	-0.2	77.8	83.8	78.3	83.7	6.6
Medium Replay	HalfCheetah	68.9 ± 2.3	67.0 ± 1.5	57.5	55.1	68.2	40.2	47.1	44.2	44.6	38.4
	Hopper	96.6 ± 7.0	97.6 ± 3.4	62.2	73.1	39.1	93.6	97.0	94.7	60.9	11.8
	Walker2D	85.0 ± 15.0	88.5 ± 4.0	49.8	56.0	69.4	49.8	88.2	73.9	81.8	11.3
Medium Expert	HalfCheetah	93.7 ± 10.5	79.3 ± 2.9	55.4	90.0	72.7	53.3	90.8	86.7	90.7	35.8
	Hopper	83.3 ± 9.1	89.5 ± 11.1	82.6	111.1	3.3	108.7	106.8	91.5	98.0	111.9
	Walker2D	68.3 ± 20.6	63.1 ± 31.3	88.8	96.1	-0.3	95.6	109.4	109.6	110.1	6.4
MuJoCo-v2 Total:		826.2 ± 33.8	785.5 ± 40.4	614.1	769.7	413.4	773.0	767.4	692.6*	698.5	300.8
AntMaze	Umaze	25.0 ± 12.0	23.8 ± 15.0	0.0	80.3	0.0	0.0	74.0	87.5	78.6	65.0
	Medium-Play	16.4 ± 17.9	5.6 ± 10.9	0.0	0.0	0.0	0.0	61.2	71.2	3.0	0.0
	Large-Play	0.0 ± 0.0	0.0 ± 0.0	0.0	0.0	0.0	0.0	15.8	39.6	0.0	0.0
	Umaze-Diverse	0.0 ± 0.0	0.0 ± 0.0	0.0	57.3	0.0	0.0	84.0	62.2	71.4	55.0
	Medium-Diverse	23.2 ± 14.2	8.4 ± 9.9	0.0	0.0	0.0	0.0	53.7	70.0	10.6	0.0
	Large-Diverse	2.4 ± 3.3	0.0 ± 0.0	0.0	0.0	0.0	0.0	14.9	47.5	0.2	0.0
AntMaze-v0 Total:		67.0 ± 14.9	37.8 ± 12.4	0.0	137.6	0.0	0.0	303.6	378.0	163.8	120.0

Table 2: Ablation of the adversarial updates for RAMBO. These results use the same rollout length for each dataset as RAMBO but with no adversarial updates (i.e. $\lambda = 0$). The scores are averaged over 5 seeds.

RAMBO (No Adversarial Training)	
MuJoCo-v2 Total: 694.4 ± 56.5	**AntMaze-v0 Total:** 45.8 ± 21.8

Table 3: Comparison between RAMBO and COMBO for the Single Transition Example. We use 20 seeds and $\pm$ captures the standard deviation over seeds. RAMBO outperforms COMBO ($p = 0.005$).

RAMBO: $\mathbf{1.49 \pm 0.05}$	**COMBO:** 1.41 ± 0.12

Ablation of Adversarial Training In Table 2 we present results for RAMBO with no adversarial updates. These results demonstrate that overall performance degrades if the adversarial training is removed. This parallels previous findings that mitigating the issue of model exploitation is crucial to obtaining strong performance in model-based offline RL. Interestingly however, for some specific datasets we obtain the best performance with no adversarial training (Appendix C.2). This suggests that a potential direction for future work could be trying to identify which types of problems do not require regularisation for a successful policy to be trained offline with model-based RL.

Comparison to COMBO We focus especially on comparing our approach to COMBO, as it is the most similar existing algorithm. We compare RAMBO and COMBO on the Single Transition toy example which is described in detail in Appendix C.1. This domain has a one-dimensional state and action space, and several distinct regions of the action space are covered by the dataset. We use this domain to investigate whether the policies optimised by RAMBO and COMBO tend to become stuck in local optima.

Table 3 compares the performance of RAMBO and COMBO on the Single Transition Example. Further analysis in Appendix C.1 shows that for this problem, the pessimistic value function updates used by COMBO create local maxima in the Q-function which are present throughout training. Policy optimisation can become stuck in these local maxima. On the other hand, the value function produced by RAMBO is initially optimistic, and pessimism is introduced into the value function *gradually* as the transition function is modified adversarially. Adversarial modification of the transition function is visualised in Appendix C.3. As a result of this gradual introduction of pessimism, we observe that the

policy produced by RAMBO is less likely to become stuck in poor local maxima, and better overall performance is obtained in Table 3. This observation may help to explain why RAMBO is able to achieve consistently strong performance relative to existing algorithms in the MuJoCo domains. Gradually increasing the level of pessimism could be a useful modification for existing offline RL algorithms to be investigated in future work.

7 Conclusion and Future Directions

RAMBO is a promising new approach to offline RL which imposes conservatism by adversarially modifying the transition dynamics of a learnt model. Our approach is theoretically justified, and achieves state-of-the-art performance on standard benchmarks.

There are a number of possible extensions to RAMBO, some of which we have already discussed. In addition, we would like to apply RAMBO to image-space domains by using deep latent variable models to compress the state space [18, 51] and adversarially perturbing the transition dynamics in the latent space representation. Another direction that we would like to investigate is the use of adversarially trained models to aid interpretability in deep RL [44] by generating imagined worst-case trajectories. Finally, we wish to investigate applying the ideas developed in this work to the online RL setting to improve robustness.

Acknowledgements

This work was supported by a Programme Grant from the Engineering and Physical Sciences Research Council (EP/V000748/1), the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services. Additionally, this project made use of time on Tier 2 HPC facility JADE2, funded by the Engineering and Physical Sciences Research Council (EP/T022205/1).

The authors would like to thank Raunak Bhattacharyya, Paul Duckworth, and Matthew Budd for their feedback on an earlier draft of this work.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[2] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. *Advances in Neural Information Processing Systems*, 34:7436–7447, 2021.

[3] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? A large-scale study. In *International Conference on Learning Representations*, 2021.

[4] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

[5] J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. Solving uncertain Markov decision processes. Technical report, 2001.

[6] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pages 619–629. PMLR, 2021.

[7] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021.

[8] Catherine Cang, Aravind Rajeswaran, Pieter Abbeel, and Michael Laskin. Behavioral priors and dynamics models: Improving performance and domain transfer in offline RL. In *Deep RL Workshop NeurIPS 2021*, 2021.

[9] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. 2022.

[10] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

[11] Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. *International Conference on Learning Representations*, 2020.

[12] Michael K Cohen and Marcus Hutter. Pessimism about unknown unknowns inspires conservatism. In *Conference on Learning Theory*, pages 1344–1373. PMLR, 2020.

[13] Sebastian Curi, Ilija Bogunovic, and Andreas Krause. Combining pessimism with optimism for robust and efficient model-based deep reinforcement learning. In *International Conference on Machine Learning*, pages 2254–2264. PMLR, 2021.

[14] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[15] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[17] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[18] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[20] Toru Hishinuma and Kei Senda. Weighted model estimation for offline model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

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

[22] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[24] Kirthevasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David Carter. Batch policy gradient methods for improving neural conversation models. *International Conference on Learning Representations*, 2017.

[25] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[27] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with Fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

[28] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

[29] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[30] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *Conference on Robot Learning*, 2021.

[31] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[32] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[33] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[34] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2020.

[35] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[36] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *International Conference on Machine Learning RL4RealLife Workshop*, 2019.

[37] Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and Stephen J Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022.

[38] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420. IEEE, 2020.

[39] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[41] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. AlgaeDICE: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[42] Xinkun Nie, Emma Brunskill, and Stefan Wager. Learning when-to-treat policies. *Journal of the American Statistical Association*, 116(533):392–409, 2021.

[43] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[44] Michael Oberst and David Sontag. Counterfactual off-policy evaluation with gumbel-max structural causal models. In *International Conference on Machine Learning*, pages 4881–4890. PMLR, 2019.

[45] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint*, 2019.

[46] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

[47] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

[48] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8522–8528. IEEE, 2019.

[49] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[50] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[51] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[52] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

[53] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. In *International Conference on Machine Learning*, pages 7953–7963. PMLR, 2020.

[54] Marc Rigter, Bruno Lacerda, and Nick Hawes. Minimax regret optimisation for robust planning in uncertain Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11930–11938. Association for the Advancement of Artificial Intelligence, 2021.

[55] Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under model mismatch. *Advances in Neural Information Processing Systems*, 30, 2017.

[56] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *International Conference on Learning Representations*, 2016.

[57] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1):109–136, 2011.

[58] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[59] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[60] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[61] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[62] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[63] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104:104366, 2021.

[64] Aviv Tamar, Shie Mannor, and Huan Xu. Scaling up robust MDPs using function approximation. In *International Conference on Machine Learning*, pages 181–189. PMLR, 2014.

[65] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[66] Masatoshi Uehara and Wen Sun. Pessimistic model-based offline reinforcement learning under partial coverage. *International Conference on Learning Representations*, 2022.

[67] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34, 2021.

[68] Yue Wang and Shaofeng Zou. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34, 2021.

[69] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[70] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

[71] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[72] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[73] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[74] Yijun Yang, Jing Jiang, Tianyi Zhou, Jie Ma, and Yuhui Shi. Pareto policy pool for model-based offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

[75] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[76] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[77] Siyuan Zhang and Nan Jiang. Towards hyperparameter-free policy selection for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

6.1 Additional Results

As with the other chapters, the entirety of the appendices for this chapter are included at the end of this document. However, here we include results from Appendix C.1 and C.3 because they give additional insight into our approach. Appendix C.1 highlights differences in our approach compared to COMBO [153], the most similar prior method. Appendix C.3 provides a visualisation of how our approach adversarially modifies the dynamics model, and how this regularises the Q-values for out-of-distribution actions.

Single Transition Example

Description In this domain, the state and action spaces are one-dimensional. The agent executes a single action, $a \in [-1, 1]$, from initial state s_0. After executing the action, the agent transitions to s' and receives a reward equal to the successor state, i.e. $r(s') = s'$, and the episode terminates.

The actions in the dataset are sampled uniformly from $a \in [-0.75, 0.7] \cup [-0.15, -0.1] \cup [0.1, 0.15] \cup [0.7, 0.75]$. In the MDP for this domain, the transition distribution for successor states is as follows:

- $s' \sim \mathcal{N}(\mu = 1,\ \sigma = 0.2)$, for $a \in [-0.8,\ -0.65]$.
- $s' \sim \mathcal{N}(\mu = 0.5,\ \sigma = 0.2)$, for $a \in [-0.2,\ -0.05]$.
- $s' \sim \mathcal{N}(\mu = 1.25,\ \sigma = 0.2)$, for $a \in [0.05,\ 0.2]$.
- $s' \sim \mathcal{N}(\mu = 1.5,\ \sigma = 0.2)$, for $a \in [0.65,\ 0.8]$.
- $s' = 0.5$, for all other actions.

The transitions to successor states from the actions in the dataset are illustrated in Figure 6.1.

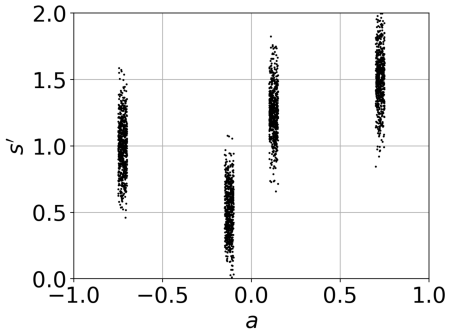


Figure 6.1: Transition data for the Single Transition Example after executing action a from s_0.

Comparison between RAMBO and COMBO To generate the comparison, we run both algorithms for 50 iterations. To choose the regularisation parameter for COMBO, we sweep over $\beta \in \{0.1, 0.2, 0.5, 1, 5\}$ and choose the parameter with the best performance. Note that 0.5, 1, and 5 are the values used in the original COMBO paper [153], so we consider lower values of regularisation than in the original paper. For RAMBO we use λ =3e-2. We used the implementation of COMBO from github.com/takuseno/d3rlpy.

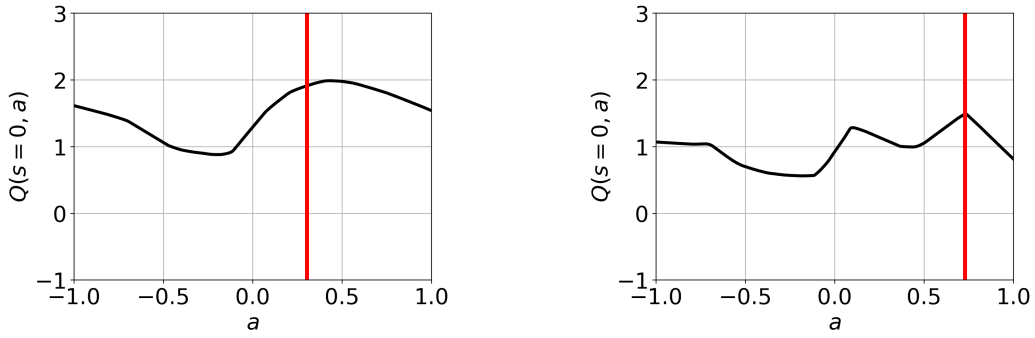
Figure 6.2 shows the Q-values produced by RAMBO throughout training. We can see that after 5 iterations, the Q-values for actions which are outside of the dataset are initially *optimistic*, and over-estimate the true values. However, after 50 iterations the Q-values for out of distribution actions have been regularised by RAMBO. The action selected by the policy after 50 iterations is the optimal action in the dataset, $a \in [0.7, 0.75]$. Thus, we see that RAMBO introduces pessimism *gradually* as the adversary is trained to modify the transitions to successor states.

For COMBO, the best performance averaged over 20 seeds was obtained for $\beta = 0.2$, and therefore we report results for this value of the regularisation parameter. Figure 6.3 illustrates the Q-values produced by COMBO throughout training (with $\beta = 0.2$). We see that at both 5 iterations and 50 iterations, the value estimates for actions outside of the dataset are highly pessimistic. In the run illustrated for COMBO, the action selected by the policy after 50 iterations is $a \in [0.1, 0.15]$, which is not the optimal action in the dataset. Figure 6.3 shows that the failure to

find an optimal action is due to the gradient-based policy optimisation becoming stuck in a local maxima of the Q-function.

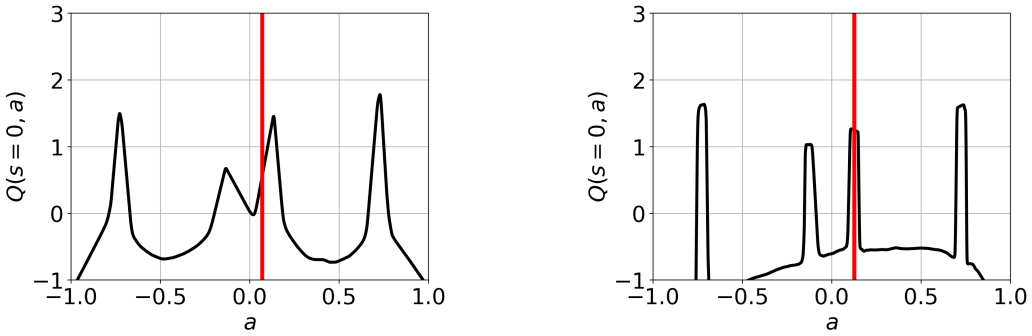
These results highlight a difference in the behaviour of RAMBO compared to COMBO. For RAMBO, pessimism is introduced gradually as the adversary is trained to modify the transitions to successor states. For COMBO, pessimism is introduced at the outset as it is part of the value function update throughout the entirety of training. Additionally, the regularisation of the Q-values for out of distribution actions appears to be less aggressive for RAMBO than for COMBO.

The results averaged over 20 seeds in Table 3 of the paper show that RAMBO consistently performs better than COMBO for this problem. This suggests that the gradual introduction of pessimism produced by RAMBO means that the policy optimisation procedure is less likely to get stuck in poor local maxima for this example. The downside of this behaviour is that it may take more iterations for RAMBO to find a performant policy. Modifying other algorithms to gradually introduce pessimism may be an interesting direction for future research.



(a) 5 iterations **(b)** 50 iterations

Figure 6.2: Q-values at the initial state during the training of RAMBO on the Single Transition Example. The red line indicates the mean action of the policy.



(a) 5 iterations **(b)** 50 iterations

Figure 6.3: Q-values at the initial state during the training of COMBO on the Single Transition Example ($\beta = 0.2$). The red line indicates the mean action of the policy.

Visualisation of Adversarially Trained Dynamics Model

To visualise the influence of training the transition dynamics model in an adversarial manner as proposed by RAMBO, we consider the following simple example MDP. In the example, the state space (S) and action space (A) are 1-dimensional with, $A = [-1, 1]$. For this example, we assume that the reward function is known and is equal to the current state, i.e. $R(s, a) = s$, meaning that greater values of s have greater expected value. The true transition dynamics to the next state s' depend on the action but are the same regardless of the initial state, s.

In Figure 6.4 we plot the data present in the offline dataset for this MDP. Note that the actions in the dataset are sampled from a subset of the action space: $a \in [-0.3, 0.3]$. Because greater values of s' correspond to greater expected value, the transition data indicates that the optimal action covered by the dataset is $a = 0.3$.

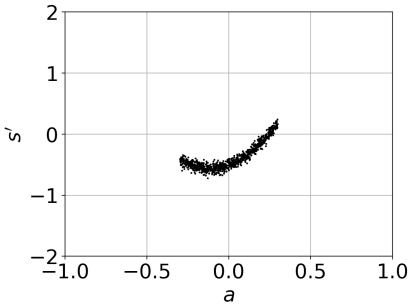
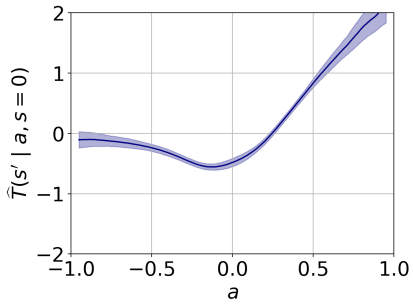
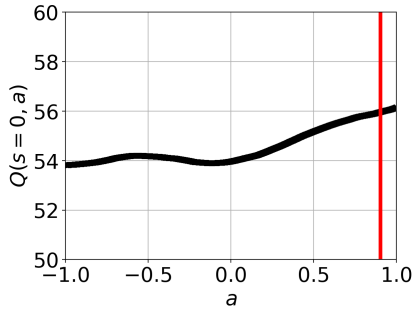


Figure 6.4: Transition data for illustrative example. The transition function is the same regardless of the initial state. The actions in the dataset are sampled from $a \in [-0.3, 0.3]$.

In Figure 6.5 we plot the output of running naïve model-based policy optimisation (MBPO) on this illustrative offline RL example. Figure 6.5a illustrates the MLE transition function used by MBPO. The transition function fits the dataset for $a \in [-0.3, 0.3]$. Outside of the dataset, it predicts that an action of $a \approx 1$ transitions to the best successor state. Figure 6.5b shows that applying a reinforcement learning algorithm (SAC) to this model results in the value function being over-estimated, and $a \approx 1$ being predicted as the best action. This illustrates that the policy learns to exploit the inaccuracy in the model and choose an out-of-distribution action.

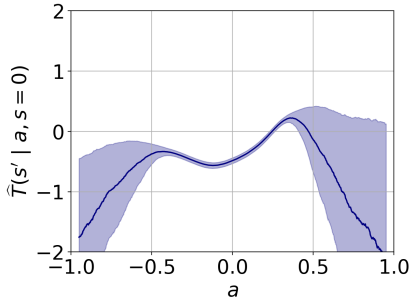


(a) Maximum likelihood estimate of the transition function, which is used by MBPO. Shaded area indicates ±1 SD of samples generated by the ensemble.

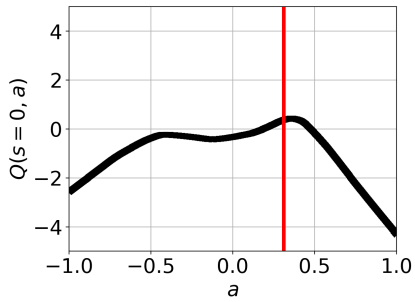


(b) Q-values after running SAC for 15 iterations. The values are significantly over-estimated. The red line indicates the mean action taken by the SAC policy.

Figure 6.5: Plots generated by running naïve MBPO on the illustrative MDP example.



(a) Model which is adversarially trained using RAMBO. Shaded area indicates ±1 SD of samples generated by the ensemble. The transition function accurately predicts the dataset for in-distribution actions, but predicts transitions to low value states for actions which are out of distribution.



(b) Q-values from SAC critic. The red line indicates the mean action taken by the policy trained using SAC, which is $a \approx 0.3$ (the best in-distribution action). Training on pessimistic synthetic transitions regularises the Q-values for out-of-distribution actions.

Figure 6.6: Output generated by running RAMBO for 15 iterations on the illustrative MDP example, using an adversary weighting of λ =3e-2.

In Figure 6.6 we show plots generated after running RAMBO on this example for 15 iterations, using an adversary weighting of $\lambda = 3\text{e-}2$. Figure 6.6a shows that the transition function still accurately predicts the transitions within the dataset. This is because choosing $\lambda \ll 1$ means that the MLE loss dominates for state-action pairs within the dataset. Due to the adversarial training, for actions $a \notin [-0.3, 0.3]$ which are out of the dataset distribution, the transition function generates pessimistic transitions to low values of s, which have low expected value.

As a result, low values are predicted for out-of-distribution actions, and the SAC agent illustrated in 6.6b learns to take action $a \approx 0.3$, which is the best action which is within the distribution of the dataset. This demonstrates that by generating pessimistic synthetic transitions for out-of-distribution actions, RAMBO enforces conservatism and prevents the learnt policy from taking state-action pairs which have not been observed in the dataset.

6.2 Limitations and Future Work

One of the limitations of RAMBO is that we found that it did not perform well on the harder AntMaze benchmarks, along with all the other model-based methods that we compared against. Understanding why model-based methods perform less well on these more difficult domains is an important question for future work. It is unclear if this is simply because the learnt model is not accurate enough, or there is some deeper limitation.

Recent work [164] suggests that model-based methods are overly aggressive and leave the dataset distribution, resulting in the robot contacting the walls of the maze. One would expect that the robust MDP formulation that we address should prevent this from happening. However, the state-action space of AntMaze domain has a greater number of dimensions than the other benchmarks (>50 vs ∼20). Adversarially training the model may be slow for high-dimension problems, as there is a large space of state-action pairs for which the model needs to be modified to generate pessimistic transitions. Because of this, it might be possible that RAMBO fails to ensure that the policy remains within the dataset distribution

for these problems. A first step to investigate this as a possible issue would be to visualise the trajectories generated by the policy, to identify whether the policy may be leaving the data distribution.

Another avenue for future work could be investigating rollouts that are generated backwards through time [164]. The advantage of a model that generates reverse rollouts is that it can be used to guarantee that that all simulated trajectories end at states contained within the dataset. This approach has been shown to improve the performance of model-based methods on the AntMaze benchmarks [164]. It would be interesting to combine the regularisation approach that we propose in RAMBO with a reverse model to see whether this improves performance.

Another limitation of our approach is that we make use of a simple model that predicts a Gaussian distribution over successor states. This type of model is suitable for the MuJoCo benchmarks that we tested on which are deterministic. However, many real-world problems are stochastic, and these stochastic outcomes may have multimodal distributions. Therefore, combining model-based offline RL algorithms with dynamics models that can more accurately predict stochastic outcomes [9] is an important direction for future work.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

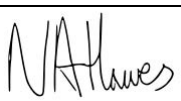
Title of Paper	RAMBO-RL: Robust adversarial model-based offline reinforcement learning.
Publication Status	□Published ☑Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Marc Rigter, Bruno Lacerda and Nick Hawes (2022). RAMBO-RL: Robust adversarial model-based offline reinforcement learning. Advances in Neural Information Processing Systems (NeurIPS).

Student Confirmation

Student Name:	Marc Rigter		
Contribution to the Paper	I developed the idea. I implemented the algorithm and ran the experiments. I led the writing in collaboration with Bruno Lacerda and Nick Hawes.		
Signature		Date	16th November 2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Nick Hawes		
Supervisor comments	I confirm that Marc made a substantial contribution to the publication, and that the description above is accurate.		
Signature		Date	22nd November 2022

This completed form should be included in the thesis, at the end of the relevant chapter.

7

One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline Reinforcement Learning

In the previous chapter we presented RAMBO, which aims to ensure that the learnt policy remains within regions covered by the dataset by generating pessimistic synthetic transitions for out-of-distribution state-action pairs. RAMBO is based on a robust MDP formulation, which optimises the policy in a worst-case instantiation of plausible MDP models. We can view this approach as achieving robustness towards the epistemic uncertainty resulting from the limited dataset. RAMBO optimises for the expected value in the worst-case MDP instantiation, and therefore is neutral towards aleatoric uncertainty.

One of the key advantages of offline RL compared to online RL is that it is more suitable for safety-critical domains where online exploration is too dangerous or costly. In many safety-critical domains in which offline RL is applicable, we want the policy to be risk-averse towards aleatoric uncertainty, and prioritise avoiding the worst outcomes that might occur due to environment stochasticity. However, the vast majority of offline RL research, including RAMBO, considers risk-neutral algorithms. This motivates the work in this chapter, where we develop a risk-sensitive approach to offline RL.

The approach in this chapter is summarised in Table 1.1. Like the previous chapter, we address offline RL so the prior knowledge about the environment is in the form of a fixed dataset. Unlike the previous chapter, which only considered epistemic

Table 1.1: Summary of the attributes of each chapter.

	Chapter 3	**Chapter 4**	**Chapter 5**	**Chapter 6**	**Chapter 7**	**Chapter 8**
Prior knowledge	MDP fully known	MDP uncertainty set	Prior over MDPs	Fixed dataset	Fixed dataset	Expert demonstrator
Considers aleatoric uncertainty	Yes	No	Yes	No	Yes	No
Considers epistemic uncertainty	No	Yes	Yes	Yes	Yes	Yes
Risk or robustness	Risk	Robustness	Risk	Robustness	Risk	N/A
Paradigm	Tabular	Tabular	Tabular	Function Approx.	Function Approx.	Function Approx.

uncertainty, in this chapter we propose a risk-sensitive approach to avoid outcomes that are plausible due to *either* epistemic or aleatoric uncertainty. As shown in Table 1.1, this is a similar core approach to that in Chapter 5. However, there are a number of differences. First, in Chapter 5, we considered the Bayes-adaptive MDP setting, where a known prior over MDPs represents the epistemic uncertainty. In this chapter we learn a model from the offline dataset which approximates the epistemic uncertainty. Second, in Chapter 5 we proposed an approach for tabular problems. In this chapter, we apply function approximation to achieve a more scalable solution. Finally, in Chapter 5 we aimed to find a policy that adapts to the true underlying MDP during execution. In this chapter, we find a Markovian policy offline and do not consider online adaptation.

Our approach in this chapter requires uncertainty estimation. Specifically, we assume that we can compute an approximation to the posterior distribution over MDP models, given the dataset. In practice, we use an ensemble of neural networks to represent the epistemic uncertainty over the MDP model. We propose to optimise the dynamic risk under this posterior distribution over models. Like Chapter 5, this means that the policy is trained to be risk-averse towards both the epistemic uncertainty over the true MDP model, in addition to the aleatoric uncertainty predicted by each plausible MDP model.

To achieve risk-aversion, we consider the dynamic risk perspective, where the desired risk measure is applied recursively at each time step. In the paper, we discuss the motivation for choosing to consider dynamic rather than static risk in this context. The dynamic risk is equivalent to the expected value under a transition distribution that is perturbed independently at each step. In our approach, to generate synthetic data for training the policy, we sample multiple candidate successor states from the ensemble. We re-weight the distribution of successor states according to the chosen risk measure, and this re-weighted distribution over transitions is added to the replay buffer for training the policy.

In the offline RL setting, risk-aversion to epistemic uncertainty discourages the policy from visiting areas not covered by the dataset, as in these areas epistemic uncertainty is high. Thus, risk-aversion to epistemic uncertainty mitigates the issue of distributional shift which is a core focus of many works on offline RL. Risk-aversion to aleatoric uncertainty ensures that poor outcomes are avoided in stochastic environments.

Our experiments show that our algorithm achieves competitive performance relative to state-of-the-art baselines on deterministic benchmarks, and outperforms existing approaches for risk-sensitive offline RL on stochastic domains.

Marc Rigter, Bruno Lacerda and Nick Hawes (2022). One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline Reinforcement Learning. *arXiv preprint arXiv:2212.00124*.

One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline Reinforcement Learning

Marc Rigter [1] **Bruno Lacerda** [1] **Nick Hawes** [1]

Abstract

Offline reinforcement learning (RL) is suitable for safety-critical domains where online exploration is too costly or dangerous. In such safety-critical settings, decision-making should take into consideration the risk of catastrophic outcomes. In other words, decision-making should be *risk-sensitive*. Previous works on risk in offline RL combine together offline RL techniques, to avoid distributional shift, with risk-sensitive RL algorithms, to achieve risk-sensitivity. In this work, we propose risk-sensitivity as a mechanism to jointly address both of these issues. Our model-based approach is risk-averse to both epistemic and aleatoric uncertainty. Risk-aversion to epistemic uncertainty prevents distributional shift, as areas not covered by the dataset have high epistemic uncertainty. Risk-aversion to aleatoric uncertainty discourages actions that may result in poor outcomes due to environment stochasticity. Our experiments show that our algorithm achieves competitive performance on deterministic benchmarks, and outperforms existing approaches for risk-sensitive objectives in stochastic domains.

1. Introduction

In safety-critical applications, online reinforcement learning (RL) is impractical due to the requirement for extensive random exploration that may be dangerous or costly. To alleviate this issue, offline (or batch) RL considers finding the best possible policy from pre-collected data. In safety-critical settings we want decision-making to be *risk-sensitive*, and take into consideration variability in the performance. However, the vast majority of offline RL research considers the expected value objective, and therefore disregards variability in performance between episodes.

One of the core issues in offline RL is avoiding distributional shift: to obtain a performant policy, we must ensure that the distribution of state-action pairs generated by the policy remains near the distribution covered by the dataset. Another way that we can view this is that we want the policy to be averse to *epistemic* uncertainty (uncertainty stemming from a lack of data). In regions not covered by the dataset, epistemic uncertainty is high and there will be large errors in the model or value function that may be exploited by the policy, resulting in the policy choosing erroneous actions.

In risk-sensitive RL, the focus is typically on finding policies that are risk-averse to environment stochasticity, i.e. *aleatoric* uncertainty. Previous approaches to risk-sensitive offline RL (Ma et al., 2021; Urpí et al., 2021) combine together two techniques: offline RL techniques, to avoid distributional shift; and risk-sensitive RL algorithms, to achieve risk-sensitivity. In other words, the issues of epistemic and aleatoric uncertainty are handled separately, using different techniques. In contrast, we propose to avoid epistemic uncertainty due to distributional shift, as well as aleatoric uncertainty due to environment stochasticity, by computing a policy that is risk-averse to *both* sources of uncertainty.

In this work we propose 1R2R, a model-based algorithm for risk-sensitive offline RL. We provide a new perspective by proposing to jointly optimise for risk-sensitivity towards both epistemic and aleatoric uncertainty. The intuition behind our approach is that we should optimise the policy so that it avoids the risk of a bad outcome *either* due to high uncertainty in the model (epistemic uncertainty) or due to randomness in the environment (aleatoric uncertainty). Unlike previous approaches to risk-sensitive offline RL (Urpí et al., 2021; Ma et al., 2021) which combine offline RL techniques with risk-sensitive optimisation, our approach modifies standard RL methods *only by considering risk*. Therefore, our approach is conceptually simpler than existing approaches. Our main contributions are:

- Proposing risk-sensitivity towards epistemic uncertainty as a mechanism to address the issue of distributional shift in offline RL.
- Introducing the first model-based algorithm for risk-sensitive offline RL, which jointly addresses epistemic and aleatoric uncertainty.

We evaluate our approach in several scenarios. Our experi-

[1]Oxford Robotics Institute, University of Oxford. Correspondence to: Marc Rigter <mrigter@robots.ox.ac.uk>.

ments show that our algorithm achieves competitive performance relative to state-of-the-art baselines on deterministic benchmarks (Fu et al., 2020), and outperforms existing approaches for risk-sensitive objectives in stochastic domains.

2. Related Work

Offline RL: Offline RL addresses the problem of learning policies from fixed datasets. Most works on offline RL are *risk-neutral*, meaning that they optimise expected value, the standard objective in sequential decision-making under uncertainty (Puterman, 2014). A key issue in offline RL is avoiding distributional shift so that the states visited by the learnt policy remain within the distribution covered by the dataset (Levine et al., 2020). Failing to mitigate this issue can lead to the policy learning to visit out-of-distribution states to exploit errors in the value function (Kumar et al., 2019) or model (Kidambi et al., 2020), and as a result, the policy may perform poorly in the real environment.

Approaches for model-free offline RL include: constraining the learnt policy to be similar to the behaviour policy (Fujimoto et al., 2019; Kostrikov et al., 2022; Wu et al., 2019; Fujimoto & Gu, 2021), conservative value function updates (Cheng et al., 2022; Kostrikov et al., 2021; Kumar et al., 2020; Xie et al., 2021), importance sampling algorithms (Liu et al., 2019; Nachum et al., 2019), or using Q-ensembles for more robust value estimates (Agarwal et al., 2020; An et al., 2021; Yang et al., 2022).

Model-based approaches learn a model of the environment and generate synthetic data from that model (Sutton, 1991) to optimise a policy via planning (Argenson & Dulac-Arnold, 2021) or RL algorithms (Yu et al., 2020; 2021). By training a policy on additional synthetic data, model-based approaches have the potential for broader generalisation (Ball et al., 2021). Approaches to preventing model exploitation include constraining the trained policy to be similar to the behaviour policy (Swazinna et al., 2021) or applying conservative value function updates (Yu et al., 2021) in the same fashion as model-free approaches. Another approach is to modify the learnt MDP by either applying reward penalties for state-action pairs with high uncertainty (Kidambi et al., 2020; Lu et al., 2022; Yang et al., 2021; Yu et al., 2020), or modifying the transition model to produce pessimistic synthetic transitions (Rigter et al., 2022b; Guo et al., 2022). Very recent work by Guo et al. (2022) samples from the model in a pessimistic manner, and is the most similar existing algorithm to our work. However, unlike our work, Guo et al. (2022) do not consider risk.

Risk-Sensitive RL: Many works have argued for risk-sensitive algorithms in safety-critical domains (Garcıa & Fernández, 2015; Majumdar & Pavone, 2020). Risk-sensitive algorithms for MDPs include approaches that adapt policy gradients to risk objectives (Chow et al., 2017; Tamar et al., 2012; 2015a;b). More recently, a number of works have utilised distributional RL (Bellemare et al., 2017; Morimura et al., 2010), which learns a distributional value function that predicts the full return distribution. Knowledge of the return distribution can then be used to optimise a risk-sensitive criterion (Dabney et al., 2018; Keramati et al., 2020; Ma et al., 2020). Most risk-sensitive RL algorithms are model-free, however model-based approaches which utilise model-predictive control have been also proposed (Webster & Flach, 2021; Zhang et al., 2020).

The main focus of research on risk-sensitive online RL is risk due to environment stochasticity (Eriksson & Dimitrakakis, 2020) (i.e. aleatoric uncertainty). However, there are a number of works that explicitly address risk due to epistemic uncertainty (Clements et al., 2020; Depeweg et al., 2018; Eriksson & Dimitrakakis, 2020; Eriksson et al., 2022; Rigter et al., 2021). To our knowledge, this is the first work to consider risk due to epistemic uncertainty as a means to mitigate distributional shift in offline RL.

Risk-Sensitive Offline RL: To our knowledge, there are two previous algorithms for risk-sensitive offline RL: ORAAC (Urpí et al., 2021) and CODAC (Ma et al., 2021). EvC (Angelotti et al., 2021) evaluates risk for risk-neutral policies in offline RL but does not propose an algorithm to generate risk-sensitive policies. To avoid distributional shift, ORAAC utilises policy constraints, while CODAC employs pessimistic value function updates to penalise out-of-distribution state-action pairs. Both algorithms learn a distributional value function which is then used to optimise a policy for a risk-sensitive objective. Thus, both of these works are model-free, and combine techniques from offline RL and distributional RL, to address the issues of distributional shift and risk-sensitivity, respectively. In this work, we provide a new perspective by proposing to optimise for risk-sensitivity towards *both* epistemic uncertainty (stemming from distributional shift) and aleatoric uncertainty (stemming from environment stochasticity). Unlike previous approaches, our approach is: a) is conceptually simpler as it does not intermix different techniques, b) does not require a distributional value function (which might be computationally demanding to train), and c) is model-based.

3. Preliminaries

An infinite-horizon MDP is defined by the tuple $M = (S, A, T, R, s_0, \gamma)$. S and A are the state and action spaces respectively, $R(s, a)$ is the reward function, $T(s'|s, a)$ is the transition function, s_0 is the initial state, and $\gamma \in (0, 1)$ is the discount factor. In this work we consider deterministic Markovian policies, $\pi \in \Pi$, which map each state to an action. In offline RL we only have access to a fixed dataset of transitions from the MDP: $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{|\mathcal{D}|}$. The goal is to find a performant policy using the fixed dataset.

Model-Based Offline RL Algorithms These algorithms use a model of the MDP to help train a policy. The learnt dynamics model, $\widehat{T}$, is typically trained via maximum likelihood estimation: $\min_{\widehat{T}} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\big[-\log \widehat{T}(s'|s,a)\big]$. A model of the reward function, $\widehat{R}(s,a)$, can also be learnt if it is unknown. This results in the learnt MDP model: $\widehat{M} = (S, A, \widehat{T}, \widehat{R}, \mu_0, \gamma)$. Thereafter, any planning or RL algorithm can be used to recover the optimal policy in the learnt model, $\widehat{\pi} = \arg\max_{\pi\in\Pi} J^{\pi}_{\widehat{M}}$. Here, J indicates the optimisation objective, which is typically the expected value of the discounted cumulative reward.

However, directly applying this approach does not perform well in the offline setting due to distributional shift. In particular, if the dataset does not cover the entire state-action space, the model will inevitably be inaccurate for some state-action pairs. Thus, naive policy optimisation on a learnt model in the offline setting can result in *model exploitation* (Janner et al., 2019; Kurutach et al., 2018; Rajeswaran et al., 2020), i.e. the policy chooses actions that the model erroneously predicts will lead to high reward. We propose to mitigate this issue by optimising a risk-averse policy.

Following previous works (Kidambi et al., 2020; Yu et al., 2021; 2020), we use model-based policy optimisation (MBPO) (Janner et al., 2019) to learn the optimal policy for $\widehat{M}$. MBPO utilises a standard off-policy actor-critic RL algorithm. However, the value function is trained using an augmented dataset $\mathcal{D}\cup\widehat{\mathcal{D}}$, where $\widehat{\mathcal{D}}$ is synthetic data generated by simulating truncated rollouts in the learnt model. To train the policy, minibatches of data are drawn from $\mathcal{D}\cup\widehat{\mathcal{D}}$, where each datapoint is sampled from the real data, $\mathcal{D}$, with probability f, and from $\widehat{\mathcal{D}}$ with probability $1-f$.

Risk Measures We denote the set of all probability distributions by $\mathcal{P}$. Consider a probability space, $(\Omega, \mathcal{F}, P)$, where Ω is the sample space, $\mathcal{F}$ is the event space, and $P \in \mathcal{P}$ is a probability distribution over $\mathcal{F}$. Let $\mathcal{Z}$ be the set of all random variables defined over the probability space $(\Omega, \mathcal{F}, P)$. We will interpret a random variable $Z \in \mathcal{Z}$ as a reward, i.e. the greater realisation of Z, the better. We denote a ξ-weighted expectation of Z by $\mathbb{E}_\xi[Z] := \int_{\omega\in\Omega} P(\omega)\xi(\omega)Z(\omega)\mathrm{d}\omega$, where $\xi : \Omega \to \mathbb{R}$ is a function that re-weights the original distribution of Z.

A *risk measure* is a function, $\rho : \mathcal{Z} \to \mathbb{R}$, that maps any random variable Z to a real number. An important class of risk measures are *coherent* risk measures. Coherent risk measures satisfy a set of properties that are consistent with rational risk assessments. A detailed description and motivation for coherent risk measures is given by Majumdar & Pavone (2020). An example of a coherent risk measure is the conditional value at risk (CVaR). CVaR at confidence level α is the mean of the α-portion of the worst outcomes.

All coherent risk measures can be represented using their dual representation (Artzner et al., 1999; Shapiro et al., 2021b). A risk measure, ρ is coherent if and only if there exists a convex bounded and closed set, $\mathcal{B}_\rho \in \mathcal{P}$, such that

$$\rho(Z) = \min_{\xi\in\mathcal{B}_\rho(P)} \mathbb{E}_\xi[Z] \tag{1}$$

Thus, the value of any coherent risk measure for any $Z \in \mathcal{Z}$ can be viewed as an ξ-weighted expectation of Z, where ξ is the worst-case weighting function chosen from a suitably defined *risk envelope*, $\mathcal{B}_\rho(P)$. For more details on coherent risk measures, we defer the reader to Shapiro et al. (2021b).

Static and Dynamic Risk In MDPs, the random variable of interest is usually the total reward received at each episode, i.e. $Z_{\mathrm{MDP}} = \sum_{t=0}^{\infty}\gamma R(s_t, a_t)$. The *static* perspective on risk applies a risk measure directly to this random variable: $\rho(Z_{\mathrm{MDP}})$. Thus, *static* risk does not take into account the temporal structure of the random variable in sequential problems, such as MDPs. This leads to the issue of "time inconsistency" (Boda & Filar, 2006). This means that if a certain action is considered less risky at one point in time, it might not be considered less risky at other points in time. This leads to non-Markovian optimal policies, and prohibits the straightforward application of dynamic programming due to the coupling of risk preferences over time.

This issue motivates *dynamic* risk measures, which take into consideration the sequential nature of the stochastic outcome, and are therefore time-consistent. Markov risk measures (Ruszczyński, 2010) are a class of dynamic risk which are obtained by recursively applying static coherent risk measures. Throughout this paper we will consider dynamic Markov risk over an infinite horizon, denoted by ρ_∞. For policy π, MDP M, and static risk measure ρ, the corresponding Markov coherent risk $\rho^\pi_\infty(M)$ is defined as

$$\rho^\pi_\infty(M) = R(s_0, \pi(s_0)) + \rho\Big(\gamma R(s_1, \pi(s_1)) + \rho\big(\gamma^2 R(s_2, \pi(s_2)) + \dots\big)\Big), \tag{2}$$

where ρ is a static coherent risk measure, and $(s_0, s_1, s_2, \dots)$ indicates random trajectories drawn from the Markov chain induced by π in M. Note that in Eq. 2 the static risk measure is evaluated at each step according to the distribution over possible successor states.

We define the risk-sensitive value function under some policy π as: $V^\pi(s) = \rho^\pi_\infty(M|s_0 = s)$. It has been shown by Ruszczyński (2010) that this value function can be found by recursively applying the *risk-sensitive* Bellman equation:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \min_{\xi\in\mathcal{B}_\rho(T(s,\pi(s),\cdot))} \sum_{s'} T(s, \pi(s), s')\cdot\xi(s')\cdot V^\pi(s'). \tag{3}$$

Likewise, the optimal risk-sensitive value, $V^*(s) = \max_{\pi \in \Pi} \rho_\infty(M|s_0 = s)$, is the unique solution to the risk-sensitive Bellman optimality equation:

$$V^*(s) = \max_{a \in A} \Big\{ R(s,a) + \gamma \min_{\xi \in \mathcal{B}_\rho(T(s,a,\cdot))} \sum_{s'} T(s,a,s') \cdot \xi(s') \cdot V^*(s') \Big\} \quad (4)$$

Thus, we can view Markov dynamic risk measures as the expected value in an MDP where the transition probabilities at each step are modified adversarially within the risk-envelope for the chosen one-step static risk measure.

4. One Risk to Rule Them All

In this section, we present *One Risk to Rule Them All* (1R2R), a new algorithm for risk-sensitive offline RL. Our approach assumes that we can compute an approximation to the posterior distribution over MDPs, given the offline dataset. 1R2R then uses an RL algorithm to train a policy using synthetic data generated from the distribution over MDP models. To achieve risk-sensitivity, the data distribution of the model rollouts is modified according to the risk envelope of the chosen risk measure. This simple modification to standard model-based RL enables our approach to avoid distributional shift, *and* generate risk-sensitive behaviour.

4.1. Problem Formulation

For ease of exposition, we assume that the reward function is known and therefore it is only the transition function that must be learned. Given the offline dataset, $\mathcal{D}$, we assume that we can approximate the posterior distribution over the MDP transition function given the dataset: $P(T \mid \mathcal{D})$. We can integrate over all possible transition functions to obtain a single expected transition function:

$$\overline{T}(s,a,s') = \int_T T(s,a,s') \cdot P(T \mid \mathcal{D}) \cdot \mathrm{d}T. \quad (5)$$

This gives rise to a single MDP representing the distribution over plausible MDPs, $\overline{M} = (S, A, \overline{T}, R, s_0, \gamma)$. This is conceptually similar to the *Bayes-Adaptive* MDP (Ghavamzadeh et al., 2015) (BAMDP), except that in the BAMDP the posterior distribution over transition functions is updated after each step of online interaction with the environment. Here, we assume that the posterior distribution is fixed given the offline dataset. To make this distinction, we refer to $\overline{M}$ as the Bayesian MDP (BMDP) (Angelotti et al., 2021). We defer the reader to Dorfman et al. (2021) and Ghosh et al. (2022) for work on offline RL in BAMDPs.

In this work, we pose the problem of offline RL as optimising the dynamic risk in the BMDP induced by the dataset.

Problem 1. Given an offline dataset, $\mathcal{D}$, static risk measure ρ, and belief over the transition dynamics given the dataset, $P(T \mid \mathcal{D})$, find the policy with the optimal dynamic risk in the corresponding Bayesian MDP $\overline{M}$:

$$\pi^* = \arg\max_\pi \rho^\pi_\infty(\overline{M}) \quad (6)$$

Motivation Equation 3 shows that dynamic risk in MDPs is equivalent to the expected value, but with the transition distribution perturbed by $\xi(s')$ at each step. The perturbation is adversarial, so transitions to low-value successor states are made more likely, and vice-versa. Consider some low-value successor state, s'. A perturbation to increase the BMDP transition probability $\overline{T}(s,a,s')$ can be written as:

$$\xi(s') \cdot \overline{T}(s,a,s') = \xi(s') \cdot \int_T T(s,a,s') \cdot P(T \mid \mathcal{D}) \cdot \mathrm{d}T$$
$$= \int_{T \in \{T \mid T(s,a,s') \neq 0\}} \xi(s') \cdot T(s,a,s') \cdot P(T \mid \mathcal{D}) \cdot \mathrm{d}T$$

Thus, we can view $\xi(s')$ as jointly perturbing the belief distribution $P(T \mid \mathcal{D})$ and the corresponding transition probability $T(s,a,s')$, for transition functions where $T(s,a,s') \neq 0$, such that the overall likelihood of transitioning to low-value successor state s' is increased. This increases the likelihood of transitioning to low-value successor states that are plausible due to either epistemic uncertainty (represented by $P(T \mid \mathcal{D})$) or aleatoric uncertainty (represented by $T(s,a,\cdot)$). Thus, the adversarial perturbation results in aversion to both sources of uncertainty.

Illustrative Example The example in Figure 1 shows a dataset drawn from a heteroskedastic transition function. Each episode consists of one step. A single reward is received, equal to the value of the successor state. We approximate $P(T \mid \mathcal{D})$ using an ensemble of neural networks, and the shaded region represents the aggregated BMDP transition function $\overline{T}(s,a,s')$. The dashed red line indicates the Q-values computed by drawing transitions from $\overline{T}(s,a,s')$. For this value function, the action with the highest Q-value is $a = 1$, which is outside of the data distribution. The solid red line indicates the *risk-sensitive* value function for the risk measure $\text{CVaR}_{0.1}$ (i.e. $\alpha = 0.1$). This is computed by drawing transitions from a uniform distribution over the worst 10% of transitions from $\overline{T}(s,a,s')$. We can see that the risk-sensitive value function penalises straying far from the dataset, where epistemic uncertainty is high. The action in the dataset with highest expected value is $a = 0.4$. However, the action $a = -0.35$ has a higher risk-sensitive value than $a = 0.4$, due to the latter having high-variance transitions (i.e. high aleatoric uncertainty). Thus, we see that choosing actions according to the risk-sensitive value function penalises actions with high uncertainty due to either epistemic or aleatoric uncertainty.

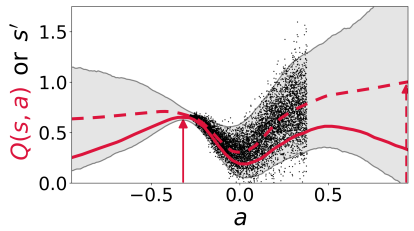


Figure 1: MDP with a single transition to a successor state s', where reward $R(s') = s'$ is received. Dataset is illustrated by black dots. Shaded region indicates ± 2 S.D. of successor states drawn from an approximation of $\overline{T}(s, a, \cdot)$. This approximation is a uniform distribution over an ensemble of probabilistic neural networks (Chua et al., 2018). Dashed red line indicates the Q-values from running MBPO (Janner et al., 2019). Solid red line indicates the risk-sensitive value function computed by reweighting the transitions generated by the ensemble according to $\text{CVaR}_{0.1}$. Arrows indicate optimal actions according to each Q-function.

4.2. Approach

Our algorithm is inspired by the model-free online RL method for aleatoric risk introduced by Tamar et al. (2015a), but we have adapted it to the model-based offline RL setting. The key idea of our approach is that to achieve risk-sensitivity, we modify the distribution of trajectories sampled in the model. Following Equation 3, we first compute the adversarial perturbation to the transition distribution:

$$\xi^* = \underset{\xi \in \mathcal{B}_\rho(\overline{T}(s,a,\cdot))}{\arg\min} \sum_{s'} \overline{T}(s, a, s') \cdot \xi(s') \cdot V^\pi(s'), \quad (7)$$

where V^π is estimated by the standard off-policy RL algorithm. When generating synthetic rollouts from the model, for each state-action pair (s, a) we sample successor states from the perturbed distribution $\xi^* \overline{T}$:

$$s' \sim \xi^*(s') \cdot \overline{T}(s, a, s') \;\; \forall s' \quad (8)$$

The transition data generated from this modified distribution is added to the synthetic dataset, $\widehat{\mathcal{D}}$. This approach modifies the sampling distribution over successor states according to the risk-sensitive Bellman equation in Equation 3. Therefore, the value function learnt by the actor-critic RL algorithm under this modified sampling distribution is equal to the risk-sensitive value function. The policy is then optimised to maximise the risk-sensitive value function.

For the continuous problems we address, it is difficult to compute the adversarially modified transition distribution by solving the optimisation problem in Equation 7 exactly. Therefore, we use sample average approximation (SAA) (Shapiro et al., 2021a) to approximate the solution to Equation 7. Specifically, we approximate $\overline{T}(s, a, \cdot)$ as a uniform distribution over m states drawn from $\overline{T}(s, a, \cdot)$. Then, we compute the optimal adversarial perturbation to this uniform distribution over successor states. Under standard regularity assumptions, the solution to the SAA converges to the exact solution as $m \to \infty$ (Shapiro et al., 2021a).

Algorithm 1 1R2R

1: **Input:** Offline dataset, $\mathcal{D}$; Static risk measure, ρ;
2: Compute belief distribution over transition models, $P(T \mid \mathcal{D})$;
3: **for** iter $= 1, \ldots, N_{\text{iter}}$:
4: **for** rollout $= 1, \ldots, N_{\text{rollout}}$, generate synthetic rollouts:
5: Sample initial state, $s \in \mathcal{D}$.
6: **for** step $= 1, \ldots, k$:
7: Sample action a according to current policy $\pi(s)$.
8: Sample m successor states, $\Psi = \{s'_i\}_{i=1}^m$ from $\overline{T}(s, a, \cdot)$.
9: Consider $\widehat{T}$, a discrete approximation to $\overline{T}$:
$$\widehat{T}(s, a, s') \approx \overline{T}(s, a, s') = \frac{1}{m}, \; \forall s' \in \Psi$$
10: Compute adversarial perturbation to $\widehat{T}$:
$$\widehat{\xi}^* = \underset{\xi \in \mathcal{B}_\rho(\widehat{T}(s,a,\cdot))}{\arg\min} \sum_{s' \in \Psi} \widehat{T}(s, a, s') \cdot \xi(s') \cdot V^\pi(s')$$
11: Sample from adversarially modified distribution:
$$s' \sim \widehat{\xi}^* \cdot \widehat{T}(s, a, \cdot)$$
12: Add $(s, a, R(s, a), s')$ to the synthetic dataset, $\widehat{\mathcal{D}}$.
13: $s \leftarrow s'$
14: *Agent update*: Update π and V^π with an actor-critic algorithm, using samples from $\mathcal{D} \cup \widehat{\mathcal{D}}$.

Algorithm Our approach is summarised in Algorithm 1. At each iteration, we generate N_{rollout} truncated synthetic rollouts which are branched from an initial state sampled from the dataset (Lines 4-6). For a given (s, a) pair we sample a set Ψ of m candidate successor states from $\overline{T}$ (Line 8). We then approximate the original transition distribution by $\widehat{T}$, a uniform distribution over Ψ, in Line 9. Under this approximation, it is straightforward to compute the worst-case perturbed transition distribution for a given risk measure in Line 10. Details of how this is computed for CVaR and Wang risk are in Appendix A. The final successor state s' is sampled from the perturbed distribution in Line 11 and the transition to this final successor state is added to $\widehat{\mathcal{D}}$.

After the synthetic rollouts, the policy and value function are updated using an off-policy actor-critic algorithm by sampling data from both the synthetic dataset and the real dataset (Line 14). Utilising samples from the real dataset means that Algorithm 1 does not precisely replicate the risk-sensitive Bellman equation (Equation 3). However, we found that additionally utilising the real data is necessary to obtain strong performance (see Appendix C.3).

Implementation Details Following a number of previous works (Guo et al., 2022; Rajeswaran et al., 2017; Yu et al., 2020) we use a uniform distribution over an ensemble of neural networks to approximate $P(T \mid \mathcal{D})$. Each model in the ensemble, T_i, outputs a Gaussian distribution over successor states and rewards: $T_i(s', r \mid s, a) =$

$\mathcal{N}(\mu_i(s,a), \Sigma_i(s,a))$. For policy training we used soft-actor critic (SAC) (Haarnoja et al., 2018) in Line 14. For policy improvement, we used the standard policy gradient that is utilised by SAC. We found this worked well in practice and meant that our approach requires minimal modifications to existing RL algorithms. Research on specialised policy gradients for risk-sensitive objectives can be found in Tamar et al. (2015a); Chow et al. (2017).

5. Experiments

In our experiments, we seek to: a) verify that 1R2R generates performant policies in deterministic environments by avoiding distributional shift, b) investigate whether 1R2R generates risk-sensitive behaviour on stochastic domains, and c) compare the performance of 1R2R against existing baselines on both stochastic and deterministic environments. To examine each of these questions, we evaluate our approach on the following domains. More information about the domains can be found in Appendix D.2. The code for the experiments can be found at github.com/marc-rigter/1R2R.

D4RL MuJoCo (Fu et al., 2020) There are three different deterministic environments representing different robots (*HalfCheetah, Hopper, Walker2D*), each with 4 datasets (*Random, Medium, Medium-Replay, Medium-Expert*).

Currency Exchange We adapt the Optimal Liquidation problem (Almgren & Chriss, 2001) to offline RL. The agent aims to convert 100 units of currency A to currency B, subject to a stochastic exchange rate, before a deadline. The dataset consists of experience from a random policy.

HIV Treatment We adapt this online RL domain (Keramati et al., 2020; Ernst et al., 2006) to the offline setting. The 6-dimensional continuous state vector represents the concentrations of various cells and viruses. The actions correspond to the dosages of two drugs. Transitions are stochastic due to the efficacy of each drug varying stochastically at each step. The reward is determined by the health outcomes for the patient, and penalises side-effects due to the quantity of drugs prescribed. The dataset is constructed in the same way as the D4RL *Medium-Replay* datasets.

Stochastic MuJoCo We modify the D4RL benchmarks by adding stochastic perturbation forces. We focus on Hopper and Walker2D as there is a risk that any episode may terminate early due to the robot falling over. We vary the magnitude of the perturbations between two different levels (*Moderate-Noise, High-Noise*). For each domain and noise level we construct *Medium, Medium-Replay*, and *Medium-Expert* datasets in the same fashion as D4RL.

Baselines We compare 1R2R against offline RL algorithms designed for risk-sensitivity (ORAAC (Urpí et al., 2021) and CODAC (Ma et al., 2021)) in addition to performant risk-neutral model-based (RAMBO (Rigter et al., 2022b)) and model-free (IQL (Kostrikov et al., 2022), TD3+BC (Fujimoto & Gu, 2021), and CQL (Kumar et al., 2020)) algorithms. For D4RL, we provide results for the -v2 datasets. Following Fu et al. (2020), the results in Tables 1, 2, and 3 are normalised to approximately between 0 and 100.

Objectives and Hyperparameters In our experiments, we evaluate the algorithms for expected performance, as well as performance for static risk objectives. For risk-sensitive algorithms ORAAC and CODAC, we set the optimisation objective to the desired objective for each domain. For all baselines, we tune hyperparameters to obtain the best online performance for the desired objective on each dataset (see Appendix D.5). Likewise, we tune the hyperparameters for 1R2R (the rollout length and risk measure parameter) to achieve the best online performance as described in Appendix B.4. Results obtained by varying the base hyperparameters of 1R2R are also included in Appendix C.3.

Versions of 1R2R We test two versions 1R2R which utilise different risk measures to re-weight the transition distribution to successor states: $\text{1R2R}_{\text{CVaR}}$ and $\text{1R2R}_{\text{Wang}}$. These risk measures are defined formally in Appendix A. We chose CVaR because it is commonly used and easy to interpret. However, CVaR disregards the best possible outcomes, and this can lead to an unnecessary loss in performance (Rigter et al., 2022a). Therefore, we also trial the Wang risk measure which is computed by reweighting the entire distribution over outcomes. Furthermore, we present additional results that we reference as the "Ablation". To generate these results we run 1R2R without modifying the transition distribution of the rollouts, i.e. without risk-aversion, making the ablation equivalent to standard model-based RL.

5.1. Results

D4RL MuJoCo For these domains, the objective is to optimise the expected performance. Thus, to generate the results in Table 1 we configure all algorithms to obtain the best expected performance. The results in Table 1 show that similar results are obtained by both $\text{1R2R}_{\text{CVaR}}$ and $\text{1R2R}_{\text{Wang}}$. The performance of both configurations of 1R2R is competitive with the risk-neutral baselines on these deterministic domains. 1R2R outperforms ORAAC and CODAC, the two existing offline RL algorithms that are able to take risk into consideration. The ablation shows that when risk-sensitivity is removed from 1R2R, for many domains the training diverges or performance degrades. This verifies that the consideration of risk in our approach is crucial for ensuring that the policy avoids distributional shift.

For some datasets (especially those for HalfCheetah), the ablation obtains similar performance to 1R2R. This shows that for some problems, standard model-based RL can be successful in the offline setting.

Table 1: Normalised expected value results for the D4RL MuJoCo-v2 benchmark. We report the average return during the last 10 iterations of training averaged over 5 seeds. Highlighted numbers indicate results within 10% of the best score. $\pm$ indicates the S.D. over seeds. "div." indicates that the value function diverged for at least one run.

		1R2R$_{\text{CVaR}}$	1R2R$_{\text{Wang}}$	Ablation	ORAAC	CODAC	RAMBO	CQL	IQL	TD3+BC
Random	HalfCheetah	36.9 ± 6.4	35.5 ± 1.7	38.6 ± 1.9	22.4 ± 1.7	31.0 ± 2.0	40.0	19.6	-	11.0
	Hopper	30.9 ± 2.3	31.8 ± 0.2	div.	10.9 ± 13.3	9.2 ± 0.7	21.6	6.7	-	8.5
	Walker2D	7.6 ± 12.2	8.1 ± 11.0	div.	2.4 ± 3.5	12.8 ± 8.9	11.5	2.4	-	1.6
Medium	HalfCheetah	74.5 ± 2.1	75.5 ± 1.2	76.9 ± 1.2	38.5 ± 4.8	62.8 ± 0.9	77.6	49.0	47.4	48.3
	Hopper	80.2 ± 15.8	64.6 ± 25.1	64.4 ± 18.5	29.6 ± 10.3	63.9 ± 1.7	92.8	66.6	66.3	59.3
	Walker2D	63.9 ± 31.8	83.4 ± 4.8	85.3 ± 0.0	45.5 ± 14.9	84.0 ± 4.0	86.9	83.8	78.3	83.7
Medium Replay	HalfCheetah	65.7 ± 2.3	65.6 ± 1.3	65.9 ± 3.4	39.3 ± 1.7	53.4 ± 1.9	68.9	47.1	44.2	44.6
	Hopper	92.9 ± 10.7	93.2 ± 8.7	55.3 ± 14.4	22.6 ± 12.1	68.8 ± 28.2	96.6	97.0	94.7	60.9
	Walker2D	92.2 ± 2.3	88.4 ± 5.0	92.7 ± 3.6	11.1 ± 5.6	73.9 ± 31.7	85.0	88.2	73.9	81.8
Medium Expert	HalfCheetah	96.0 ± 6.0	94.5 ± 1.9	87.8 ± 6.3	24.4 ± 7.6	76.7 ± 4.9	93.7	90.8	86.7	90.7
	Hopper	81.6 ± 22.9	89.6 ± 17.8	88.6 ± 13.4	4.1 ± 2.8	87.6 ± 6.4	83.3	106.8	91.5	98.0
	Walker2D	90.9 ± 5.9	78.1 ± 7.9	div.	42.8 ± 9.2	112.0 ± 2.3	68.3	109.4	109.6	110.1

Currency Exchange We use this toy example to qualitatively examine the trade-off between risk and expected performance achieved by 1R2R. In Figure 2, we observe that the risk-neutral algorithm, RAMBO, obtains the best performance in expectation. However, the return distribution for RAMBO includes a considerable tail of poor outcomes as evidenced by the static CVaR$_{0.1}$ performance for RAMBO. For 1R2R, we observe that as we use a more risk-averse dynamic risk measure, the resulting behaviour becomes more risk-averse. For more risk-averse parameter settings, the number of poor outcomes in the tail of the return distribution is reduced, at the expense of worse performance in expectation. These results verify that by adjusting the dynamic risk measure, we are able to obtain different levels of risk-sensitivity towards aleatoric uncertainty with 1R2R.

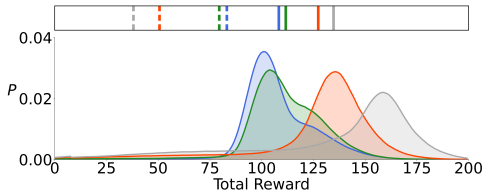


Figure 2: Unnormalized return distributions for the Currency Exchange domain, aggregated over 10 seeds. Results are generated by 1R2R with the following risk measures for the dynamic risk: CVaR$_{0.5}$ (blue); CVaR$_{0.7}$ (green); CVaR$_{0.9}$ (red). The return distribution for risk-neutral RAMBO is grey. Solid vertical bars indicate average return, dashed vertical bars indicate the mean of the worst 10% of returns (i.e. static CVaR$_{0.1}$).

HIV Treatment As this is a healthcare setting, we assume that the desired behaviour is to be highly risk-averse. Therefore, we assume that the objective is to optimise static CVaR$_{0.02}$, and configure all algorithms to obtain the best performance for this objective. Results for this objective, as well as the expected performance for each algorithm are in Table 2. We observe that 1R2R obtains the best performance for the static CVaR$_{0.02}$ objective. RAMBO obtains similar, yet slightly worse performance. The return distributions for each algorithm in Appendix C.2 show that

Table 2: Normalized performance on the HIV Treatment domain. The results are generated from 500 evaluation episodes at each of the last 10 iterations of training, and averaged over 10 seeds. Highlighted numbers indicate results within 10% of the best score. $\pm$ indicates S.D. over seeds.

Algorithm	Static CVaR$_{0.02}$	Expected Value
1R2R$_{\text{CVaR}}$	43.7 ± 1.5	73.5 ± 1.9
1R2R$_{\text{Wang}}$	42.3 ± 3.9	68.9 ± 7.5
Ablation	33.9 ± 17.9	56.7 ± 30.1
ORAAC	0.1 ± 0.1	3.7 ± 6.7
CODAC	9.1 ± 1.3	58.7 ± 1.5
RAMBO	42.0 ± 1.5	73.2 ± 1.2
CQL	12.5 ± 2.3	63.5 ± 1.8
IQL	0.0 ± 0.0	27.1 ± 4.9
TD3+BC	19.1 ± 5.5	70.2 ± 1.2
SAC (Online)	38.4 ± 9.6	100.0 ± 18.2

the policies obtained by 1R2R generate qualitatively different behaviour than RAMBO. This suggests that 1R2R generates a different policy from RAMBO, that is more tailored towards risk-aversion. TD3+BC obtains similar performance to 1R2R for the expected value objective, but performs much worse for static CVaR$_{0.02}$. This shows that strong expected performance is not sufficient to ensure good performance in the worst outcomes. ORAAC and CODAC, the risk sensitive-algorithms which are configured to optimise static CVaR$_{0.02}$, both perform poorly. A possible explanation for the poor performance of these algorithms is that it is difficult to estimate the tail of the return distribution using the limited offline dataset via distributional RL.

For the static CVaR$_{0.02}$ objective, 1R2R and RAMBO both outperform a SAC agent that has been trained online. This is despite only having access to considerably less training data that was generated by a sub-optimal SAC agent. This verifies that it is possible for offline RL algorithms to recompose existing data to obtain better risk-sensitive performance.

Stochastic MuJoCo Following Urpí et al. (2021); Ma et al. (2021), we assume that the objective for these domains is to optimise static CVaR$_{0.1}$. The performance of each algorithm for static CVaR$_{0.1}$ is in Table 3, and the performance

Table 3: Normalised static $\text{CVaR}_{0.1}$ results for Stochastic MuJoCo. We report static $\text{CVaR}_{0.1}$ evaluated over the last 10 iterations of training, and averaged over 5 seeds. Highlighted numbers indicate results within 10% of the best score. $\pm$ indicates the S.D. over seeds. "div." indicates that the value function diverged for at least one run.

		1R2R$_{\text{CVaR}}$	1R2R$_{\text{Wang}}$	Ablation	ORAAC	CODAC	RAMBO	CQL	IQL	TD3+BC
Medium	Hopper (Mod)	36.9 ± 5.1	37.3 ± 1.9	35.7 ± 2.9	10.2 ± 3.9	33.6 ± 0.1	37.8 ± 8.3	33.5 ± 0.1	33.0 ± 1.4	33.4 ± 0.1
	Walker2D (Mod)	29.9 ± 8.7	28.3 ± 4.2	7.7 ± 11.6	17.7 ± 4.6	26.8 ± 0.8	8.0 ± 0.5	32.0 ± 1.7	22.7 ± 2.3	22.9 ± 11.5
	Hopper (High)	38.6 ± 3.0	42.0 ± 1.2	40.2 ± 1.2	18.5 ± 10.0	36.8 ± 0.3	36.5 ± 7.3	38.7 ± 0.8	35.8 ± 0.9	37.4 ± 0.5
	Walker2D (High)	19.4 ± 12.6	19.5 ± 5.2	div.	6.3 ± 7.4	32.2 ± 3.6	19.7 ± 0.7	43.1 ± 2.7	16.7 ± 3.5	48.2 ± 3.4
Medium Replay	Hopper (Mod)	43.9 ± 11.7	37.1 ± 7.6	33.6 ± 2.8	7.4 ± 1.3	32.1 ± 4.7	35.1 ± 5.7	34.1 ± 2.2	-0.6 ± 0.0	24.2 ± 2.3
	Walker2D (Mod)	30.1 ± 7.2	28.0 ± 7.1	29.7 ± 2.2	4.7 ± 2.5	23.3 ± 10.4	30.9 ± 5.7	25.4 ± 1.9	1.4 ± 0.6	28.8 ± 1.0
	Hopper (High)	51.4 ± 8.1	46.1 ± 7.9	30.5 ± 10.9	14.7 ± 15.2	38.9 ± 0.8	36.4 ± 8.6	50.9 ± 2.8	7.3 ± 5.8	35.5 ± 2.0
	Walker2D (High)	46.0 ± 10.1	28.8 ± 5.2	30.9 ± 14.3	1.1 ± 0.8	43.1 ± 14.4	42.2 ± 10.4	15.6 ± 6.1	-1.2 ± 0.3	29.9 ± 6.8
Medium Expert	Hopper (Mod)	65.4 ± 31.7	45.7 ± 35.7	32.6 ± 26.3	14.9 ± 9.6	44.1 ± 10.9	32.0 ± 12.6	60.6 ± 11.7	75.0 ± 29.7	54.2 ± 3.3
	Walker2D (Mod)	34.8 ± 6.5	38.5 ± 8.9	div.	14.7 ± 3.6	28.1 ± 12.5	29.9 ± 14.2	72.3 ± 9.0	24.5 ± 1.2	68.4 ± 2.7
	Hopper (High)	35.5 ± 5.7	39.3 ± 4.6	30.5 ± 7.4	9.1 ± 9.4	47.1 ± 1.7	38.5 ± 2.1	47.1 ± 2.1	32.6 ± 4.3	51.0 ± 1.3
	Walker2D (High)	10.8 ± 4.3	11.7 ± 5.2	div.	4.2 ± 6.2	26.7 ± 16.5	1.9 ± 1.4	55.5 ± 5.5	28.0 ± 8.8	60.2 ± 8.9

for expected value is in Table 6 in the appendix. We observe that 1R2R$_{\text{CVaR}}$ performs the best for static $\text{CVaR}_{0.1}$ in the *Medium* and *Medium-Replay* datasets. For these benchmarks, 1R2R$_{\text{Wang}}$ performs worse than 1R2R$_{\text{CVaR}}$. A possible explanation is that because the Wang risk measure makes use of the entire distribution over outcomes, 1R2R$_{\text{Wang}}$ trains the policy on some over-optimistic data. On the other hand, 1R2R$_{\text{CVaR}}$ disregards the most optimistic outcomes, resulting in more robust performance. For the ablation, either training diverges or the performance degrades relative to 1R2R$_{\text{CVaR}}$ for most datasets.

1R2R$_{\text{CVaR}}$ substantially outperforms ORAAC and CODAC, the prior risk-sensitive algorithms. Interestingly, while IQL performs well for the deterministic benchmarks, it performs very poorly for the *Medium-Replay* stochastic benchmarks. This suggests that we should not rely on deterministic domains to benchmark algorithms. CQL and TD3+BC both obtain fairly strong performance. They easily outperform 1R2R$_{\text{CVaR}}$ on the multimodal *Medium-Expert* datasets, where 1R2R performs poorly. These results, in addition to the results for CODAC and ORAAC, indicate that there is still considerable progress to be made towards developing better risk-sensitive offline RL algorithms.

Finally, the expected value results in Table 6 show that 1R2R$_{\text{CVaR}}$ performs slightly less well for expected performance than static $\text{CVaR}_{0.1}$, compared to the baselines. However, there is a strong correlation between the results for expected value and the results for static $\text{CVaR}_{0.1}$. This indicates that for these domains, there is only a modest tradeoff between risk and expected performance.

6. Discussion and Conclusion

Static vs Dynamic Risk We chose to base our algorithm on dynamic risk, which evaluates risk recursively at each step, rather than static risk which evaluates risk across episodes. The dynamic risk perspective is arguably more suitable for avoiding distributional shift, as it enforces aversion to uncertainty at each time step, thus preventing the policy from leaving the data-distribution at any step. It is also more amenable to our model-based approach, where we modify the transition distribution of synthetic data at each step. Applying this same approach to the static risk setting would likely require modifying the distribution over entire trajectories (Chow et al., 2017; Tamar et al., 2015b). However, sampling full length trajectories is not desirable in model-based methods due to compounding modelling error (Janner et al., 2019). The disadvantage of dynamic risk is that it is difficult to choose the risk measure to apply and it can lead to overly conservative behaviour (Gagne & Dayan, 2021). Furthermore, it is unclear how to interpret and evaluate performance for dynamic risk. For this reason, we evaluated static risk objectives in our experiments.

Modelling Stochasticity The transition distribution for each model is Gaussian. This means that our implementation is not suitable for domains with multimodal dynamics, such as those proposed by Urpí et al. (2021). To improve our approach, an obvious next step is therefore to utilise better models of stochasticity that can handle general multimodal distributions (Hafner et al., 2021).

Epistemic vs Aleatoric Uncertainty Our approach is agnostic to the source of uncertainty. This is motivated by the intuition that poor outcomes should be avoided due to either source of uncertainty. A direction for future work is to use composite risk measures (Eriksson et al., 2022), which enable the aversion to each source of risk to be adjusted independently. This might be important for achieving strong performance even if the estimation of each source of uncertainty is poorly calibrated.

Conclusion We have proposed 1R2R, a model-based algorithm for risk-sensitive offline RL that is: a) simpler than existing approaches to risk-sensitive offline RL which intermix different techniques, and b) performs well empirically. As discussed, there many promising avenues for future work.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Almgren, R. and Chriss, N. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40, 2001.

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. *Advances in Neural Information Processing Systems*, 34:7436–7447, 2021.

Angelotti, G., Drougard, N., and Chanel, C. P. C. Exploitation vs caution: Risk-sensitive policies for offline learning. *arXiv preprint arXiv:2105.13431*, 2021.

Argenson, A. and Dulac-Arnold, G. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

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

Ball, P. J., Lu, C., Parker-Holder, J., and Roberts, S. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pp. 619–629. PMLR, 2021.

Bao, W. and Liu, X.-y. Multi-agent deep reinforcement learning for liquidation strategy analysis. *arXiv preprint arXiv:1906.11046*, 2019.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. PMLR, 2017.

Boda, K. and Filar, J. A. Time consistent dynamic risk measures. *Mathematical Methods of Operations Research*, 63(1):169–186, 2006.

Cheng, C.-A., Xie, T., Jiang, N., and Agarwal, A. Adversarially trained actor critic for offline reinforcement learning. 2022.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

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

Clements, W. R., Van Delft, B., Robaglia, B.-M., Slaoui, R. B., and Toth, S. Estimating risk and uncertainty in deep reinforcement learning. *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. In *International Conference on Machine Learning*, pp. 1096–1105. PMLR, 2018.

Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., and Udluft, S. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pp. 1184–1193. PMLR, 2018.

Dorfman, R., Shenfeld, I., and Tamar, A. Offline meta reinforcement learning – identifiability challenges and effective data collection strategies. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Eriksson, H. and Dimitrakakis, C. Epistemic risk-sensitive reinforcement learning. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020.

Eriksson, H., Basu, D., Alibeigi, M., and Dimitrakakis, C. Sentinel: Taming uncertainty with ensemble based distributional reinforcement learning. In *Uncertainty in Artificial Intelligence*, pp. 631–640. PMLR, 2022.

Ernst, D., Stan, G.-B., Goncalves, J., and Wehenkel, L. Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 667–672. IEEE, 2006.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Gagne, C. and Dayan, P. Two steps to risk sensitivity. *Advances in Neural Information Processing Systems*, 34: 22209–22220, 2021.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Geramifard, A., Dann, C., Klein, R. H., Dabney, W., and How, J. P. RLPy: a value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16(1):1573–1578, 2015.

Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Ghosh, D., Ajay, A., Agrawal, P., and Levine, S. Offline RL policies should be trained to be adaptive. In *International Conference on Machine Learning*, pp. 7513–7530. PMLR, 2022.

Guo, K., Shao, Y., and Geng, Y. Model-based offline reinforcement learning with pessimism-modulated dynamics belief. *Advances in Neural Information Processing Systems*, 2022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *ICLR*, 2021.

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

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

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL: Model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 21810–21823, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with Fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy Q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

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

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Off-policy policy gradient with state distribution correction. *International Conference on Machine Learning RL4RealLife Workshop*, 2019.

Lu, C., Ball, P., Parker-Holder, J., Osborne, M., and Roberts, S. J. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022.

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

Ma, Y., Jayaraman, D., and Bastani, O. Conservative offline distributional reinforcement learning. *Advances in Neural Information Processing Systems*, 34:19235–19247, 2021.

Majumdar, A. and Pavone, M. How should a robot assess risk? towards an axiomatic theory of risk in robotics. In *Robotics Research*, pp. 75–84. Springer, 2020.

Maller, R. A., Müller, G., and Szimayer, A. Ornstein–uhlenbeck processes and extensions. *Handbook of financial time series*, pp. 421–437, 2009.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. Parametric return density estimation for reinforcement learning. *Conference on Uncertainty in Artificial Intelligence*, 2010.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. AlgaeDICE: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Popko, W., Wächter, M., and Thomas, P. Verification of continuous time random walk wind model. In *The 26th International Ocean and Polar Engineering Conference*. OnePetro, 2016.

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

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. EPOpt: Learning robust neural network policies using model ensembles. *ICLR*, 2017.

Rajeswaran, A., Mordatch, I., and Kumar, V. A game theoretic framework for model based reinforcement learning. In *International Conference on Machine Learning*, pp. 7953–7963. PMLR, 2020.

Rigter, M., Lacerda, B., and Hawes, N. Risk-averse Bayes-adaptive reinforcement learning. *Advances in Neural Information Processing Systems*, 34:1142–1154, 2021.

Rigter, M., Duckworth, P., Lacerda, B., and Hawes, N. Planning for risk-aversion and expected value in MDPs. *International Conference on Automated Planning and Scheduling*, 2022a.

Rigter, M., Lacerda, B., and Hawes, N. RAMBO-RL: Robust adversarial model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2022b.

Rockafellar, R. T. and Uryasev, S. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

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

Shapiro, A., Dentcheva, D., and Ruszczynski, A. *Lectures on stochastic programming: modeling and theory*, chapter 5. SIAM, 2021a.

Shapiro, A., Dentcheva, D., and Ruszczynski, A. *Lectures on stochastic programming: modeling and theory*, chapter 6. SIAM, 2021b.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Swazinna, P., Udluft, S., and Runkler, T. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104: 104366, 2021.

Tamar, A., Di Castro, D., and Mannor, S. Policy gradients with variance related risk criteria. *International Conference on Machine Learning*, 2012.

Tamar, A., Chow, Y., Ghavamzadeh, M., and Mannor, S. Policy gradient for coherent risk measures. *Advances in Neural Information Processing Systems*, 28, 2015a.

Tamar, A., Glassner, Y., and Mannor, S. Optimizing the CVaR via sampling. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015b.

Urpí, N. A., Curi, S., and Krause, A. Risk-averse offline reinforcement learning. *International Conference on Learning Representations*, 2021.

Wang, J., Li, W., Jiang, H., Zhu, G., Li, S., and Zhang, C. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34, 2021.

Wang, S. S. A class of distortion operators for pricing financial and insurance risks. *Journal of risk and insurance*, pp. 15–36, 2000.

Webster, S. R. and Flach, P. Risk sensitive model-based reinforcement learning using uncertainty guided planning. *arXiv preprint arXiv:2111.04972*, 2021.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Xie, T., Cheng, C.-A., Jiang, N., Mineiro, P., and Agarwal, A. Bellman-consistent pessimism for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Yang, R., Bai, C., Ma, X., Wang, Z., Zhang, C., and Han, L. RORL: Robust offline reinforcement learning via conservative smoothing. *Advances in Neural Information Processing Systems*, 2022.

Yang, Y., Jiang, J., Zhou, T., Ma, J., and Shi, Y. Pareto policy pool for model-based offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. COMBO: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Zhang, J., Cheung, B., Finn, C., Levine, S., and Jayaraman, D. Cautious adaptation for reinforcement learning in safety-critical settings. In *International Conference on Machine Learning*, pp. 11055–11065. PMLR, 2020.

7.1 Limitations and Future Work

We chose to base our approach on dynamic risk, which evaluates risk recursively at each step, rather than static risk, which evaluates risk across episodes. As discussed in the paper, our motivation for this decision was that: a) dynamic risk is more suitable for avoiding distributional shift, as it enforces aversion to uncertainty at each time step, thus discouraging the policy from leaving the data-distribution at any step; and b) optimising dynamic risk is more straightforward for our model-based algorithm, as it only requires the transition function to be modified independently at each step. The main disadvantage of this approach is that it is difficult to interpret what the algorithm ultimately optimises. This makes it unclear how we should choose the risk measure to apply. In our work, we circumvented this issue by treating the risk measure as a hyperparameter, which we tuned for the best performance for a static risk objective. However, it would be preferable for the user to be able to choose a desired static risk objective upfront, and avoid this tuning procedure. This is especially important for offline RL where hyperparameter tuning is a key issue: ideally the hyperparameters should be chosen offline without evaluation in the real environment.

A key direction for future work is improving the models of uncertainty. We used a uniform distribution over a neural network ensemble, where the transition distribution for each model is Gaussian. This prohibits our approach from being suitable for problems with multimodal stochasticity. Furthermore, neural network ensembles may poorly estimate the uncertainty over the MDP dynamics [159]. Our risk-sensitive approach is agnostic to the source of uncertainty, motivated by the intuition that the worst plausible outcomes should be avoided, regardless of the source of uncertainty. However, this approach may fail if the uncertainty estimates are poorly calibrated. For example, if the epistemic uncertainty is underestimated, the agent may not be sufficiently discouraged from entering regions of the state-action space whether the model is highly inaccurate.

To overcome this issue, one approach is to adjust the level of aversion to each source of uncertainty, so that it is possible to obtain strong performance in the

presence of poorly calibrated uncertainty estimates. This could be achieved by making use of *composite* risk measures [180]. The disadvantage of this approach is that it would increase the number of hyperparameters, as there would be a separate hyperparameter related to each source of uncertainty. Thus, adapting 1R2R to use composite risk measures may result in an even greater reliance on online hyperparameter tuning.

A final shortcoming of our work is that in some of the domains we created, particularly the Stochastic MuJoCo domains, there is a strong correlation between average performance and the performance for the risk-sensitive objective. This might be because the optimal risk-sensitive policy in these domains is similar (or the same) as the optimal expected value policy. This made it difficult to assess whether our algorithm generates different behaviour than risk-neutral algorithms. To confirm that our algorithm generates risk-sensitive behaviour, we created the Currency Exchange domain, which has a distinct tradeoff between risk and expected performance. However, this is a toy domain which is not representative of real world problems. A useful direction for future work would be to develop more realistic benchmarks for safety-critical domains, which are tailored towards evaluating risk-sensitivity and/or the robustness of offline RL algorithms. This would be analogous to Safety Gym [181], a set of benchmarks that evaluate safe exploration in online RL.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline Reinforcement Learning.
Publication Status	□Published □ Accepted for Publication □Submitted for Publication ☑Unpublished and unsubmitted work written in a manuscript style
Publication Details	Marc Rigter, Bruno Lacerda and Nick Hawes (2022). One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline Reinforcement Learning. Preprint.

Student Confirmation

Student Name:	Marc Rigter		
Contribution to the Paper	I developed the idea. I implemented the algorithm and ran the experiments. I led the writing in collaboration with Bruno Lacerda and Nick Hawes.		
Signature		Date	16th November 2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Nick Hawes		
Supervisor comments I confirm that Marc made a substantial contribution to the publication, and that the description above is accurate.			
Signature		Date	22nd November 2022

This completed form should be included in the thesis, at the end of the relevant chapter.

8

A Framework for Learning from Demonstration with Minimal Human Effort

Offline RL, which we have addressed in the previous two chapters, is one approach to circumventing the extensive exploration required by online RL. In this final piece of work, we consider the alternative perspective of utilising expert demonstrations to quickly train a policy. In this chapter, we fine-tune the policy using online RL, but little exploration is required as our approach predominantly relies on demonstrations to direct the policy towards good actions. Most works on learning from demonstration assume that a fixed batch of demonstrations is provided upfront. Our work proposes a novel interactive setting where demonstrations are requested as necessary, in an effort to minimise the human input required during training.

We focus on robotics tasks with a binary outcome of success or failure. Failing to complete a task requires the system to be reset by a human operator. Examples of such failures could include the robot getting stuck during an inspection task, or dropping a package in a factory setting. Furthermore, we assume that the robot is being trained in a real environment, such as a factory, and therefore we cannot control the order in which tasks appear (i.e. we cannot perform curriculum learning).

Table 1.1: Summary of the attributes of each chapter.

	Chapter 3	Chapter 4	Chapter 5	Chapter 6	Chapter 7	Chapter 8
Prior knowledge	MDP fully known	MDP uncertainty set	Prior over MDPs	Fixed dataset	Fixed dataset	Expert demonstrator
Considers aleatoric uncertainty	Yes	No	Yes	No	Yes	No
Considers epistemic uncertainty	No	Yes	Yes	Yes	Yes	Yes
Risk or robustness	Risk	Robustness	Risk	Robustness	Risk	N/A
Paradigm	Tabular	Tabular	Tabular	Function Approx.	Function Approx.	Function Approx.

Our goal in this piece of work is to develop a framework for interactively training the policy, while minimising the human effort required. We assume that human effort is required to generate expert demonstrations, and to reset the system from failures. To model this problem, we assume that there is a cost for generating a demonstration, as well as a cost for resetting the robot from failure. The goal is to minimise the expected cost associated with human effort accumulated over many episodes.

Table 1.1 summarises this chapter in relation to the other chapters in this thesis. In this chapter, the source of knowledge used to train the policy is the access to the expert demonstrator. We seek to optimise the human-related cost in expectation, and therefore this approach does not consider aleatoric uncertainty. Our approach does consider epistemic uncertainty. The epistemic uncertainty stems from the fact that we do not know which tasks the current policy performs well at, as opposed to the tasks where the policy would benefit from more demonstrations. As indicated by Table 1.1, our approach in this work is neither risk-averse nor robust to this epistemic uncertainty. Instead, we use "optimism in the face of uncertainty" to explore areas of high epistemic uncertainty to obtain a greater understanding of the performance of the current policy. Furthermore, the approach that we present in this chapter is entirely model-free, in contrast to all of the other chapters of this thesis.

In our proposed approach, we formulate the problem as a contextual multi-armed bandit. The context for the problem is represented by the initial state of the

environment for each episode, and this encodes the difficulty of the current task. For example, the context could be the size and position of a package which is to be manoeuvred. Based on experience from previous episodes, we train a classifier that attempts to predict the probability that the current policy will succeed at the task, as well as provide an estimate of the uncertainty in this prediction. To capture the fact that the policy is changing over time, we only use a window of recent experiences to make this prediction. Furthermore, we also allow for the possibility that there are multiple autonomous controllers to choose from, including the policy currently being trained.

For each new episode, we use the classifier to estimate the failure cost associated with each controller, given the context for that episode. Based on this estimate, we use an upper-confidence bound algorithm to decide whether autonomous control should be selected or whether a human demonstration should be requested. The result is that demonstrations are requested for cases where the policy does not perform well, and the system chooses to rely on the autonomous policy in situations where it reliably succeeds. In situations where there is high uncertainty over the performance of the policy, the upper-confidence bound algorithm encourages trialling the policy to better understand it's current performance.

We test our approach on a navigation domain and a robot arm manipulation domain. Our experiments verify that our approach reduces the human cost incurred over many episodes relative to baselines. We also validate our approach on a real-world robotic task.

Marc Rigter, Bruno Lacerda, and Nick Hawes (2020). A framework for learning from demonstration with minimal human effort. *IEEE Robotics and Automation Letters* (RAL).

A Framework for Learning from Demonstration with Minimal Human Effort

Marc Rigter[1], Bruno Lacerda[1], and Nick Hawes[1]

***Abstract*—We consider robot learning in the context of shared autonomy, where control of the system can switch between a human teleoperator and autonomous control. In this setting we address reinforcement learning, and learning from demonstration, where there is a cost associated with human time. This cost represents the human time required to teleoperate the robot, or recover the robot from failures. For each episode, the agent must choose between requesting human teleoperation, or using one of its autonomous controllers. In our approach, we learn to predict the success probability for each controller, given the initial state of an episode. This is used in a contextual multi-armed bandit algorithm to choose the controller for the episode. A controller is learnt online from demonstrations and reinforcement learning so that autonomous performance improves, and the system becomes less reliant on the teleoperator with more experience. We show that our approach to controller selection reduces the human cost to perform two simulated tasks and a single real-world task.**

***Index Terms*—Learning from demonstration, human-centered robotics, telerobotics and teleoperation**

I. Introduction

INTEGRATING demonstrations with Reinforcement Learning (RL) has been applied to a number of difficult problems in sequential decision making and control. Examples include Atari games [1] and manipulation tasks [2] [3]. In most settings, it is assumed that there is a fixed set of demonstration data. In this work, we are interested in *shared autonomy*, where control may switch back and forth between a human teleoperator, and autonomous control. This is common in many domains in which fully autonomous systems are not yet viable, and human intervention is required to increase system capability, or recover from failures (e.g. [4] [5] [6]). In shared autonomy it is desirable to reduce the burden on the human operator by only handing control to the human in situations where autonomous performance is poor, and only handing control to the robot when autonomous performance is good. The autonomous experience and human demonstrations may be used to improve autonomous performance so that the system becomes less reliant on the human for future tasks.

Elements of this problem have been addressed before. Optimally switching control in shared autonomy has been approached using Markov Decision Process (MDP) planning [7] [8]. However, most planning approaches assume a known model for the performance of the human and autonomous system, which may not be available in reality, or may change over time. Requesting human input has been approached with the *ask for help* framework, where the agent asks for a human demonstration if uncertainty is above a hand-tuned threshold [9] [10] [11] [12]. Unlike the ask for help approaches, we seek to explicitly minimise a cost function associated with human exertion, and unlike the planning approaches we do not assume models are known a priori, and learn a control policy online.

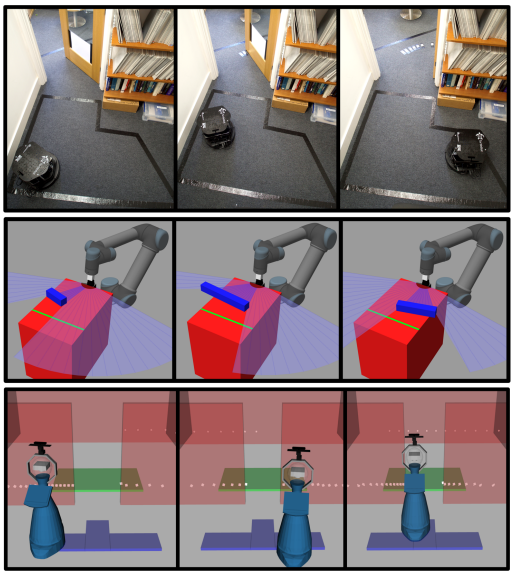


Fig. 1: Reprensentative initial configurations in our evaluation domains: real-world navigation (top), block pushing (middle) and simulated navigation (bottom).

This work considers episodic problems with a binary outcome of success or failure. We assume that there is a cost for asking the human for a demonstration, and a cost for failing an episode. These costs represent the time required for the human to perform a demonstration, or to recover the robot from failure. Motivating examples include robot navigation, where a human assists the robot when it is stuck [5], and manipulation, where a human recovers items dropped by the robot [13]. For each episode, the system must decide whether to ask for a human demonstration, or give control to one of the available autonomous controllers. We assume that an autonomous controller is learnt online as the system gathers more experience. We also allow for additional autonomous controllers, as in many domains there exist pre-programmed controllers which can be utilised to reduce operator workload.

Manuscript received: September, 10, 2019; Revised December, 18, 2019; Accepted January, 22, 2020.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments.

[1]Marc Rigter, Bruno Lacerda, and Nick Hawes are with the Oxford Robotics Institute, University of Oxford, United Kingdom (email: {mrigter, bruno, nickh} @robots.ox.ac.uk)

Digital Object Identifier (DOI): see top of this page.

The contributions of this paper are a general framework for this problem, and a specific instantiation of the framework. In the framework, controller choice is formulated as a contextual Multi-Armed Bandit (MAB). In our instantiation, we use continuous correlated beta processes to estimate the MAB probabilities. A control policy is learnt online from demonstrations plus RL so that performance improves and the system becomes increasingly autonomous. We evaluate our approach on three domains (Figure 1), including experiments on a real robot, and show empirically that our approach reduces the total human cost relative to other approaches.

II. Related Work

Previous research has combined human input with RL to guide the learning process. In supervised actor-critic RL [14], the action applied at each time step is a weighted sum of the actions suggested by a human supervisor and by the learner. In interactive RL, the reward signal is given by a human [15].

Other approaches modify the control input of the human to improve performance whilst maintaining the control authority of the operator. Reddy et al. [16] apply deep RL to learn an approximate Q-function. The action applied to the system is the action near the teleoperator input with highest value. [17] uses Model Predictive Control (MPC) to determine the optimal control from a known model, with a cost for deviating from the user control inputs. [18] also uses MPC, but learns the model from data collected from human-machine interaction. All of the above approaches require that the human operator is present at all times, while we are interested in methods which reduce the input required from the operator.

The ask for help framework introduced in [9] only asks for human input when the learner is uncertain. In [9], the agent nominally performs tabular Q-learning. If all actions in a state have a similar Q-value, the agent is deemed uncertain, and asks the human which action to take. A similar idea is applied in the Deep Q-Network (DQN) setting in [12], where the loss of the DQN is used to gauge uncertainty. Chernova and Veloso [11] consider policies defined by a support-vector machine classifier. For states near a decision boundary of the classifier, a demonstration is requested. In [10], Gaussian process policies are learnt from demonstrations. A demonstration is requested if the variance of the action output is too high. In contrast to our work, all of these approaches require a hand-tuned threshold for uncertainty.

In *predictive advising* [19], the teacher learns to predict the action that the learner will take in a given state. In a new state, if the predicted action does not equal the "correct" action known by the teacher, the teacher chooses the action taken. This method would be difficult to apply to continuous action spaces where it is unclear whether an action is correct.

MDP planning has been applied to the problem of optimally switching control between a human and agent. In [20], the authors find Pareto-optimal MDP policies which trade off minimising a cost function for operator effort, and maximizing the probability of success of satisfying a temporal logic specification. Such a cost representing human inconvenience is often referred to as "bother" cost [21]. Wray et al. [7] consider that the human and autonomous system have different known capabilities, and generate plans such that the system never enters states where the entity in control cannot act. These approaches assume a model is known for the human and autonomous agent. Lacerda et al. [22] plan over an MDP with transition probabilities learnt from executing each transition many times [23]. This approach does not require prior knowledge, but is limited to long-term deployment in the same environment. The authors use human intervention to recover from failures [5], but do not plan for human control.

Probabilistic policy reuse [24] assumes access to a library of baseline policies which may be reused, overriding the actions of the current policy with some probability. The algorithm maintains an average of the return from reusing each policy. Higher returns increase the likelihood a policy is chosen. This approach determines which baseline policy is most useful for learning the task. In our work, we consider variable controller performance throughout the state space, and choose the controller depending on the initial state.

A comparison can be made between our work and hierarchical RL [25] which uses *options*. An option is a temporally extended action, similar to the choice of controller in our work. In hierarchical RL, the policy which chooses options is also learnt with RL algorithms. In contrast, we consider controller choice as a contextual MAB and use uncertainty estimates to guide exploration towards promising choices.

In contrast to previous work, we both explicitly minimise a cost function associated with the human workload, and do not assume prior knowledge of models of performance.

III. Preliminaries

A. Multi-Armed Bandits

In a Multi-Armed Bandit (MAB), at each episode an agent must choose from a finite set Φ of arms, with unknown reward distributions r_ϕ for choosing each arm $\phi \in \Phi$. The objective is to maximise the cumulative reward received. Algorithms which solve the MAB problem must balance exploration to gather information about the arms with exploiting the best known arm. In this work, we consider an extension of the MAB that is *non-stationary* and *contextual.*

In a contextual MAB [26], at episode k the agent also receives information about the "context" in the form of a state s_k, which may inform the agent about the reward distribution of each arm for episode k. Over many trials, algorithms for the contextual MAB estimate the mean and variance of the reward for arm $\phi \in \Phi$, given state s_k where $\mu(\phi, s_k) = \mathbb{E}[r_\phi | s_k]$, and $\sigma(\phi, s_k)^2 = var[r_\phi | s_k]$. One effective approach to arm selection is to use an Upper Confidence Bound (UCB) algorithm [26]:

$$\phi_k = \arg\max_{\phi \in \Phi} \Big(\mu(\phi, s_k) + \alpha \sigma(\phi, s_k) \Big), \tag{1}$$

where α trades off exploration versus exploitation.

Our contextual MAB may also be non-stationary with the reward distribution changing between episodes (eg. due to learning). A simple approach to address this is to use a *sliding window* and only use the m most recent trials to estimate the distributions in Equation 1 [27].

B. Continuous Correlated Beta Processes

We wish to choose a function approximator to predict the probabilities required in the MAB. To estimate probabilities in the range [0,1], the output of a Gaussian Process (GP) can be passed through a logistic function to form a Logistic GP (LGP). However, for LGPs, the posterior can only be computed approximately, and computation is cubic with the size of the dataset. A simpler alternative is the Continuous Correlated Beta Process (CCBP), proposed in [28]. Like LGPs, CCBPs are a nonparametric function approximator with a model of uncertainty, and a range of [0,1], but have computation time linear with the number of data points.

Let S be a continuous state space and $\mathcal{B} = \{B_s \mid s \in S\}$ the space of Bernoulli trials indexed by states of S, with unknown success probability $p(s) = \Pr(B_s = 1)$. The outcome, $o \in \{0, 1\}$ of each trial may either be successful, denoted 1, or unsuccessful, denoted 0. The distribution over $p(s)$ after $\alpha(s)$ outcomes of success and $\beta(s)$ outcomes of failure is given by a beta distribution:

$$\Pr(p(s)) = Beta(\alpha(s), \beta(s)) \propto p(s)^{\alpha(s)-1}(1 - p(s))^{\beta(s)-1} \tag{2}$$

To obtain accurate values for $p(s)$ over the entire state space, we would need to observe many outcomes at each s, which is not possible in continuous space. In a CCBP we assume $p(s)$ is a smooth function, such that the Bernoulli trials are correlated, and experience can be shared between them. A kernel function, $K(s, s') \in [0, 1]$ is used to determine the extent to which experience from trial B_s should be shared with $B_{s'}$. Consider the set $O = \{B_{s_0} = o_0, B_{s_1} = o_1, \ldots, B_{s_T} = o_T\}$ of observed outcomes $o_0, \ldots, o_T$ after running $T + 1$ different trials. The posterior beta distribution for an experiment B_s is given by:

$$\begin{aligned}\Pr(p(s)|O) \propto\ & p(s)^{\alpha(s)-1+\sum_{t=0}^{T} o_t K(s_t,s)} \\ & \times (1 - p(s))^{\beta(s)-1+\sum_{t=0}^{T}(1-o_t)K(s_t,s)}.\end{aligned} \tag{3}$$

C. Deep Deterministic Policy Gradients

In an RL problem, an agent must learn to act by receiving a reward signal corresponding to performing an action $a \in A$ at a state $s \in S$. At discrete time step t, the agent takes action a_t according to a policy, $\pi : S \to A$. We define the return $G_t = \sum_{i=t}^{T} \gamma^{(i-t)} r_i$ where T is the horizon, $\gamma < 1$ is the discount factor, and r_i is the reward received at time step i. We consider episodic problems in which T is the end of the episode. The goal of RL is to find π which maximises the expected return, $J = \mathbb{E}_\pi[G_t]$.

Deep Deterministic Policy Gradients (DDPG) [29] is an off-policy, model-free, RL algorithm capable of utilizing neural network function approximators. DDPG maintains a state-action value function, or critic, $Q(s, a)$, with parameters θ^Q, and a policy, or actor, $\pi(s)$, with parameters θ^π. Additionally, a replay buffer R of recent transitions experienced is maintained containing tuples of (s_t, a_t, r_t, s_{t+1}).

The algorithm alternates between collecting experience by acting in the environment, and updating the actor and critic. For exploration, noise is added to the actions chosen by the actor during training: $a_t = \pi(s_t) + \mathcal{N}$, where $\mathcal{N}$ is a noise process. During each update step, a minibatch of N samples is taken from R to update the actor and critic functions. The critic parameters, θ_Q, are updated to minimise the loss

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta_Q))^2, \tag{4}$$

where the targets y_i are computed from the Bellman equation:

$$y_i = r_i + \gamma Q(s_{i+1}, \pi(s_{i+1}|\theta^\pi)|\theta^Q). \tag{5}$$

The actor is updated using the deterministic policy gradient:

$$\nabla_{\theta^\pi} J = \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\pi(s_i)} \nabla_{\theta^\pi} \pi(s|\theta^\pi)|_{s=s_i} \tag{6}$$

Intuitively, this update changes the actor parameters to produce actions with a higher Q-value as judged by the critic. The target values in Equation 5 are usually computed using separate networks for the actor and critic whose weights, $\theta^{\pi'}$ and $\theta^{Q'}$, are an exponential average over time of θ^π and θ^Q respectively. This is necessary to stabilise learning.

IV. Methodology

We assume that our system must perform an episodic task. Each episode k has an initial state, $s_{k,0} \in S_0$, and a binary outcome, $o_k \in \{0, 1\}$, of success for reaching a goal state $s_g \in S_g$, or failure. For each episode a controller must be chosen. The controller may be human teleoperation, which we denote C_h, or one of the n available autonomous controllers, $C_{a,i}$. The autonomous controllers may be pre-programmed, or learnt online. We assume that there is a cost, c_d, per human demonstration representing the human time to give a demonstration, and a cost c_f for failure, representing the human time required to recover the robot from failure. For the k^{th} episode, the system chooses a controller, $C_k \in \mathcal{C}$, where $\mathcal{C} = \{C_h, C_{a,i}, \ldots, C_{a,n}\}$. The objective is to minimise the cumulative cost $\sum c_h$, defined as the sum of the demonstration and failure costs over the episodes.

In the next subsection, we describe our general approach for controller selection to minimise the cumulative cost. Controller selection is considered as a contextual MAB, and an upper-confidence bound algorithm is applied to choose the controller at each episode. We then describe how the CCBP can be used for the performance prediction aspect of our framework, and how a controller can be learnt using DDPG with demonstrations so that the autonomous performance improves, making the system less reliant on the human. An open-source implementation of our framework is available[1].

A. A General Approach for Controller Selection

Our approach to controller selection is illustrated in Figure 2. We assume the existence of a performance prediction function, that for each controller $C_i \in \mathcal{C}$, and an initial state $s_{k,0}$ returns the probability of success for that controller for the episode, $\hat{p}(s_{k,0}, C_i)$, and the standard deviation of that probability

[1] github.com/ori-goals/lfd-min-human-effort

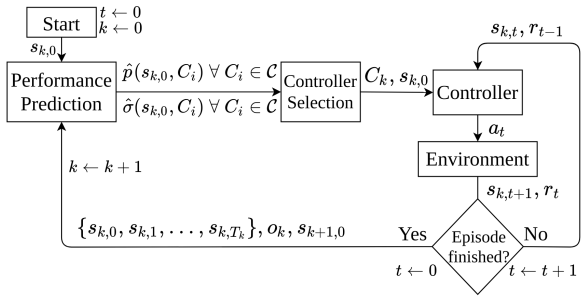


Fig. 2: High-level diagram of framework.

estimate, $\hat{\sigma}(s_{k,0}, C_i)$. One possibility for this function, based on CCBPs, will be described in Section IV-B.

These estimates are then used to perform controller selection. The controller selection problem is considered as a contextual MAB, in which we seek to minimise the total human cost. To choose a controller for episode k, we select

$$C_k = \arg\min_{C_i \in \mathcal{C}} \hat{c}_h(s_{k,0}, C_i), \tag{7}$$

where

$$\hat{c}_h(s_{k,0}, C_i) = \begin{cases} (1 - \hat{p}(s_{k,0}, C_i) - \alpha\hat{\sigma}(s_{k,0}, C_i))c_f + c_d & \text{if } C_i = C_h \\ (1 - \hat{p}(s_{k,0}, C_i) - \alpha\hat{\sigma}(s_{k,0}, C_i))c_f & \text{otherwise.} \end{cases} \tag{8}$$

Intuitively, $\hat{c}_h(s_{k,0}, C_i)$ is an optimistic lower bound on the cost for using controller C_i. This is analogous to UCB algorithms [26], but here we minimise cost, rather than maximise reward, and include the demonstration cost.

The chosen controller executes actions in the environment, until the episode ends. During each step, the environment produces a reward, r_t, which may be used to train the controllers. At the end of the episode, the states encountered during the episode, and the outcome of success or failure, o_k, are fed back to improve the performance prediction function. The environment chooses a new starting state, $s_{k+1,0}$, for the next episode. This framework is agnostic to the method for performance prediction, or the controllers available. In the following subsections, we describe the use of the CCBP for performance prediction, and DDPG with demonstrations as an approach to learning one of the controllers.

B. Performance Prediction with the CCBP

Under the assumption that the probability of succeeding at an episode for a given controller varies smoothly throughout the state space, we use a CCBP to estimate this probability, and its uncertainty. During episode k which is under the control of C_k, we observe a series of states from the continuous state space $\{s_{k,0}, s_{k,1}, \ldots, s_{k,T_k}\}$. At the end of the episode we observe outcome $o_k^{C_k}$, where we introduce the superscript to indicate the controller for that episode. We consider the outcomes of episodes as outcomes of correlated Bernoulli trials. In episode k, we observe the same outcome for each state, $O_k = \{B_{s_{k,0}} = o_k^{C_k}, B_{s_{k,1}} = o_k^{C_k}, \ldots, B_{s_{k,T_k}} = o_k^{C_k}\}$. After $n+1$ episodes, the entire set of observed outcomes is defined as $O = \bigcup_{k=0}^{n} O_k$. For a given controller C_i, we define the set of outcomes associated with that controller as $O^{C_i} = \{B_{s_{k,j}} = o_k^{C_k} \in O \mid C_k = C_i\}$

Given a previously unvisited state, s, we can calculate a beta distribution over $\Pr(B_s = 1 \mid O^{C_i})$ using a CCBP as defined in Equation 3, for each $C_i \in \mathcal{C}$. In the CCBP, we initialise $\alpha(s) = \alpha_0^{C_i}$, and $\beta(s) = \beta_0^{C_i}$. This encodes a prior assumption about the probability of success of C_i in the absence of information from correlated outcomes. The contributions from the outcomes in O^{C_i} are then incorporated per Equation 3 to produce the new beta distribution. For the success probability estimate, we take the expected value of this beta distribution: $\hat{p}(s, C_i) = \mathbb{E}[\Pr(B_s = 1 \mid O^{C_i})]$. The standard deviation of the probability estimate is also calculated from the beta distribution: $\hat{\sigma}(s, C_i)^2 = var[\Pr(B_s = 1 \mid O^{C_i})]$.

For the kernel function we chose the Gaussian kernel

$$K(s, s') = e^{-\frac{||s-s'||^2}{l}}, \tag{9}$$

where l is the length scale hyperparameter which determines the scale over which the correlation between outcomes decays with distance in the state space. The Gaussian kernel was chosen as it is a common choice for modelling smooth data.

In our framework, we also allow for a controller to be learnt online from demonstrations and RL. In this case, the performance of the controller is constantly changing, and so the performance prediction should not be influenced by out of date data. We follow the approach taken for GP value functions [30] and non-stationary MABs [27], and only consider recent data in the CCBP. This is based on the assumption that the policy changes gradually, such that recent outcomes approximate the performance of the current controller. Specifically, if C_l is a controller which is learnt online, then the outcomes we consider are those from the set $O^{C_l} = \{B_{s_{k,j}} = o_k^{C_k} \in O \mid C_k = C_l,\ k > n - m\}$, where n is the current episode number, and m is a constant. That is, O^{C_l} is the set of outcomes from episodes controlled by C_l in a sliding window of the most recent m episodes. If there are many recent episodes from other controllers, O^{C_l} is populated with fewer outcomes. This increases $\hat{\sigma}(s, C_l)$, indicating our reduced certainty in our probability estimates for C_l after many recent demonstrations.

To simplify our experiments we assume that the human teleoperation controller is always successful, that is: $\hat{p}(s, C_h) = 1$ and $\hat{\sigma}(s, C_h) = 0$ for all $s \in S$. However, this assumption is not necessary to apply our framework, and the human performance could also be predicted with a CCBP.

C. DDPG with Demonstrations

To provide an autonomous controller which is learnt online from demonstrations and RL, we adapt the approach from [3] to incorporate demonstrations into DDPG using behaviour cloning and a *Q-filter*. As DDPG is an off-policy algorithm, we add experience from all $C_i \in \mathcal{C}$ into the replay buffer R. Experience from successful episodes by controllers $C_i \neq C_l$ are also added into a demonstration replay buffer R_D.

An update is performed after each time step of any episode. During each update, we draw N experience tuples from R, and N_D tuples from R_D. The following Behaviour Cloning (BC) loss, L_{BC}, is applied only to the tuples from R_D:

$$L_{BC} = f \sum_{i=1}^{N_D} ||\pi(s_i|\theta^\pi) - a_i||^2, \tag{10}$$

$$f = \mathbb{1}_{Q(s_i,a_i) > Q(s_i,\pi(s_i)) - \epsilon|Q(s_i,\pi(s_i))|}, \tag{11}$$

where $\epsilon \geq 0$, a_i is the demonstrator action, and $\mathbb{1}_A$ is 1 if A is true and 0 otherwise. The Q-filter in Equation 11 results in the behaviour cloning loss not being applied when the critic determines that the actor action is significantly better than the demonstrator action. This prevents the actor being tied to the demonstrations if it discovers better actions. This Q-filter is modified from [3] by including the ϵ term, which when $\epsilon > 0$ reduces the experiences filtered out from the BC loss.

The gradient applied to θ^π is:

$$\lambda_1 \nabla_{\theta^\pi} J - \lambda_2 \nabla_{\theta^\pi} L_{BC}, \tag{12}$$

where λ_1 and λ_2 weight the policy gradient and behaviour cloning contributions to the update.

V. Experiments

We evaluated our approach in three domains. We assumed the availability of three controllers: $\mathcal{C} = \{C_h, C_b, C_l\}$. C_h was a human teleoperating the robot with a gamepad. C_b was a pre-programmed baseline controller capable of succeeding at some of the episodes. C_l was a control policy learnt online using the method outlined in Section IV-C. In all domains, the maximum length of an episode was 50 time steps, and the sliding window was $m = 50$ episodes. We set $\alpha_0^{C_i}$, $\beta_0^{C_i}$ to define a prior assumption that $\hat{p}(s_{k,0}, C_l) = \hat{p}(s_{k,0}, C_b) = 0.8$, and $\hat{\sigma}(s_{k,0}, C_l) = \hat{\sigma}(s_{k,0}, C_b) = 0.35$, in the absence of any observed outcomes ($\alpha_0^{C_i} = 0.245$, $\beta_0^{C_i} = 0.0612$). To break ties in Equation 7, we prioritised $C_b > C_l > C_h$.

A. Domains

1) Block Pushing: A 6-DOF arm was simulated in Gazebo [31] to push a block from a random initial configuration along a table into a goal region (see Figure 1). The state space is 32 dimensional, consisting of 30 depth measurements from a stationary laser, and 2 measurements specifying the current location of the end effector. The action space is 2 dimensional, specifying a horizontal change in position of the end effector. In each new episode, the arm position is reset, and a block is spawned onto a 0.3m×0.45m table. The block has a 0.04m×0.04m cross section and can be one of three lengths: 0.12, 0.2, or 0.3m. The initial position and orientation of the block is varied by up to ±10cm relative to the centre of the table, and ±20°. An episode is a failure if the block falls off the table, or the time limit is exceeded. An episode is a success if the entirety of the block is pushed past the green line (Figure 1). The baseline controller, C_b, was a partially trained policy using the method in Section IV-C and a sample of the recorded human demonstrations. Training of C_b was terminated when the policy was capable of ≈70% of the tasks. The reward r_t used to train the learnt policy consisted of a small dense reward for moving the block forwards, and a large reward for reaching the goal region.

2) Simulated Navigation: A Scitos G5[2] robot was simulated in the Morse simulator [32]. The task for the robot was to navigate from a starting location, through a gap, to a goal region. The starting location was randomly selected within a region. The start and goal regions are shown in Figure 1. The starting orientation was randomly varied ±10°. The width of the gap was varied according to a normal distribution with a mean of 0.83m (the standard wheelchair accessible door width in the United Kingdom), a standard deviation of 0.15m, and a lower bound of 0.68m. This lower bound is the minimum width at which the 0.62m robot can be reliably teleoperated through the gap. The position of the other walls was randomly varied by ±0.25m. The state space was a 40 dimensional laser-depth measurement, and the action space was 2 dimensional, specifying a distance to move and a change in orientation. The baseline controller, C_b, was a controller from the ROS navigation stack with the parameters fine-tuned for the Scitos G5 from the STRANDS project[3]. The reward r_t used to train the learnt policy consisted of a large reward for reaching the goal region, and 0 otherwise.

3) Real-World Navigation: The real-world navigation experiments used a Turtlebot 2[4]. The task was to navigate from a starting location and through a partially closed door. Possible starting locations are indicated by the region marked on the floor in Figure 1. To configure a new episode, the ROS navigation stack was used to navigate the robot to a random position within this region, and a random orientation varied up to ±15°. The door was set randomly to one of 8 possible positions indicated by markings on the floor. Three possible door positions are shown in Figure 1. The state space is 33 dimensional, comprising of 30 depth measurements given by the Kinect sensor on the Turtlebot, and 3 measurements specifying the estimated position and orientation of the robot given by a particle filter. The particle filter estimate is required because the Kinect has a narrow field of view, meaning that the depth measurements alone are insufficient to characterise the state. The action space was 2 dimensional, consisting of linear and angular velocity. The reward was a sparse reward for passing through the door, and 0 otherwise.

B. Length Scale Hyperparameter Estimation

The hyperparameter l was estimated from data in all domains. We initialised a policy π capable of completing approximately half of the tasks. The policy attempted 50 random tasks to populate O. The policy then executed another 50 episodes from new random initial states, which were not added to O. The likelihood of the outcomes of the second set of tasks was calculated using the CCBP for a range of values for l. The maximum likelihood estimate was then used for the CCBP kernel for both C_l and C_b. This value was $l = 4.1$ in the simulated navigation domain, $l = 2.1$ in the real-world navigation domain, and $l = 0.72$ in the block pushing domain. We leave estimating this parameter online to future work.

[2] metralabs.com/en/mobile-robot-scitos-g5/
[3] github.com/strands-project/strands_movebase
[4] clearpathrobotics.com/turtlebot-2-open-source-robot/

C. Training Details

To train C_l, we used Adam [33] with learning rate of 10^{-3} for Q, and 10^{-4} for π. The discount factor γ was 0.99. We used $N = 128$, $N_D = 64$, $\lambda_1 = 1$, $\lambda_2 = 10$. The value for λ_2 was chosen for best empirical performance after 100 demonstrations and 500 RL episodes out of $\{1, 10, 100\}$ in block pushing. The function approximators for π and Q are fully-connected neural networks with two hidden layers of 64 and 32 weights respectively, and ReLU activation. The output activation for π is tanh, and the value is rescaled into the action range. The exploration noise $\mathcal{N}$ is an Ornstein-Uhlenbeck process with $\sigma = 0.2$, $\theta = 0.15$. The noise was decayed by $0.998^{n_{C_l}}$, where n_{C_l} is the number of episodes completed by C_l. The Q-filter tolerance was $\epsilon = 0.02$ unless stated otherwise.

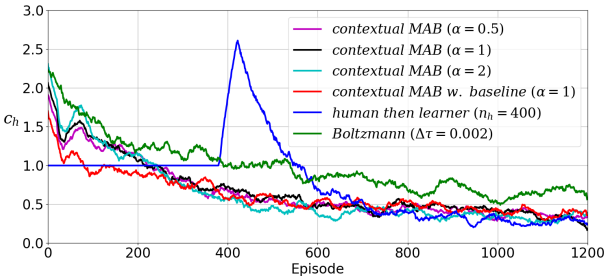
D. Simulated Human Cost Evaluation

We evaluated the cumulative human cost of several methods in simulation. The costs were $c_d = 1$ and $c_f = 5$ for both simulated domains. n random initial states were generated for each simulated domain. In box pushing, $n = 1200$ and in simulated navigation, $n = 500$. A human demonstration for each of the n possible initial states was recorded. Each run of the experiment consisted of randomly initialising the networks for C_l, and then performing a number of episodes (400 or 1200 in simulated navigation or box pushing respectively). In each run, the initial state for each episode was chosen in a random order from the set of possible initial states. In each run, the same recorded human demonstration was used for each initial state. We compared several methods, where α indicates the value used in the MAB: *contextual MAB* (α) is our approach with $\mathcal{C} = \{C_h, C_l\}$, *contextual MAB with baseline* (α) is our approach with $\mathcal{C} = \{C_h, C_b, C_l\}$, *baseline only* always uses C_b and *RL only* always uses C_l. The method *human then learner* (n_h), first executes n_h human demonstrations with C_h before switching to C_l. For *Boltzmann* ($\Delta\tau$), $\mathcal{C} = \{C_h, C_b, C_l\}$, and we adapted the method in [24]. The controller is chosen according to:

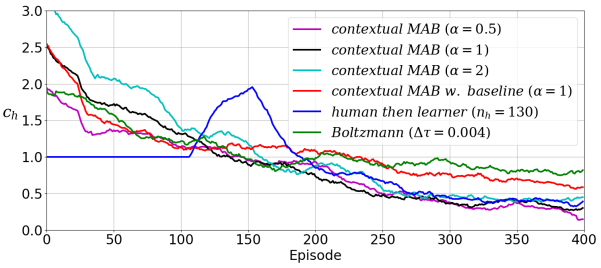
$$Pr(C_i) = \frac{e^{\tau(c_f - \bar{c}_h(C_i))}}{\sum_{C_j \in \mathcal{C}} e^{\tau(c_f - \bar{c}_h(C_j))}} \tag{13}$$

where $\bar{c}_h(C_i)$ is the average human cost incurred from the episodes in O^{C_i}. Temperature parameter τ is initially 0, and incremented by $\Delta\tau$ every episode.

Results: The average human cost over 15 runs of each method is plotted in Figure 3. For clarity, the cost is plotted using a sliding window average over 40 episodes. The average cumulative cost and episodes performed by each controller is displayed in Tables I and II. In block pushing, *contextual MAB* outperforms *human then learner* in total cost for all values of α and n_h. Smaller α values corresponded with asking for more demonstrations. The method *contextual MAB with baseline* further reduces the cost by using the baseline controller when the learnt policy is poor. Despite the fact that *baseline only* does not perform well, this is possible as *contextual MAB with baseline* tends to use the baseline for episodes it will likely succeed at. The *Boltzmann* method performs comparatively poorly for each of the values of $\Delta\tau$.



(a) Block pushing



(b) Simulated Navigation

Fig. 3: Mean cost from 15 runs of each method. For clarity, c_h is plotted using a sliding window average over 40 episodes.

TABLE I: Cumulative cost over 15 runs of each method in the box pushing domain in the format: mean (std dev.)

Method	Total Cost	Human Episodes	Baseline Episodes
context. MAB ($\alpha = 0.5$)	790 (117)	376 (76)	-
context. MAB ($\alpha = 1$)	820 (119)	332 (73)	-
context. MAB ($\alpha = 2$)	758 (167)	220 (67)	-
human then learner ($n_h = 200$)	1029 (90)	200	-
human then learner ($n_h = 300$)	959 (133)	300	-
human then learner ($n_h = 400$)	934 (94)	400	-
context. MAB w. baseline ($\alpha = 1$)	746 (75)	197 (32)	236 (35)
Boltzmann ($\Delta\tau = 0.001$)	1266 (74)	392 (15)	327 (25)
Boltzmann ($\Delta\tau = 0.002$)	1178 (94)	364 (30)	277(35)
Boltzmann ($\Delta\tau = 0.004$)	1204 (188)	500 (221)	244 (67)
baseline only	1613 (18)	-	1200
RL only	4947 (39)	-	-

TABLE II: Cumulative cost over in the simulated navigation domain in the format: mean (std dev.)

Method	Total Cost	Human Episodes	Baseline Episodes
context. MAB ($\alpha = 0.5$)	358 (51)	144 (33)	-
context. MAB ($\alpha = 1$)	366 (48)	110 (30)	-
context. MAB ($\alpha = 2$)	438 (53)	67 (16)	-
human then learner ($n_h = 70$)	517 (147)	70	-
human then learner ($n_h = 100$)	409 (75)	100	-
human then learner ($n_h = 130$)	349 (71)	130	-
context. MAB w. baseline ($\alpha = 1$)	433 (72)	57 (19)	118 (47)
Boltzmann ($\Delta\tau = 0.001$)	499 (46)	137 (9)	133 (11)
Boltzmann ($\Delta\tau = 0.002$)	505 (50)	155 (17)	147 (24)
Boltzmann ($\Delta\tau = 0.004$)	473 (56)	153 (27)	162 (26)
baseline only	443 (14)	-	400
RL only	1811 (46)	-	-

In the simulated navigation domain, *contextual MAB* performs better with a smaller value for α. This is likely because this simpler task can be learnt from demonstrations alone and no RL experience. Therefore, conservatively relying on more human demonstrations enables a good policy to be learnt

quickly and results in less cost. This also explains the strong performance of *human then learner* ($n_h = 130$). This suggests that our approach is better suited to more complex tasks which take longer to learn, and require RL experience to train a policy to complete the task. The method *contextual MAB with baseline* has less cost near the start of the runs, but performs worse overall. This is likely because the agent is slower to learn a good policy using the dissimilar demonstrations from the baseline controller and human in the simulated navigation domain. In the box pushing domain, the baseline and human give similar demonstrations because the baseline is a policy learnt from human demonstrations. Therefore this is not an issue in the box pushing domain.

E. Effect of Varying Costs

Some experiments were repeated in the block pushing domain to analyse the effect of the cost values. The demonstration cost, $c_d = 1$ for all experiments. We used $c_f \in \{3, 5, 7\}$ for *contextual MAB* and *human then learner* with a comparable number of demonstrations.

Results: Our approach incurs less cost than *human then learner* for all values of c_f (Table III). Increasing the value of c_f increases the number of human demonstrations used and vice versa. This is because if the cost for failure is higher the estimated probability of success must be higher for the algorithm to choose the autonomous controller.

TABLE III: Cost over 15 runs of 1200 episodes for block pushing with different c_f values in format: mean (std dev.)

Method	c_f	**Total Cost**	**Human Episodes**
context. MAB ($\alpha = 1$)	3	629 (82)	281 (52)
human then learner ($n_h = 300$)	3	715 (57)	300
context. MAB ($\alpha = 1$)	5	820 (119)	332 (73)
human then learner ($n_h = 300$)	5	959 (133)	300
context. MAB ($\alpha = 1$)	7	1067 (121)	421 (74)
human then learner ($n_h = 400$)	7	1154 (122)	400

F. Limited Demonstrations Evaluation

We evaluated the quality of the learnt policy when the number of human demonstrations is kept to a strict limit. After the demonstration budget was used up, C_l was used from then onwards. To evaluate policy performance, after each episode we performed an additional evaluation episode which always used C_l. The initial state for each evaluation episode was sampled randomly. The success rate of C_l in the evaluation episodes is plotted in Figure 4 for block pushing, where the limit on human demonstrations was 120. We additionally compared *human then learner* with $\epsilon = 0$ for the Q-filter.

Results: Early in training, *human then learner* outperforms *contextual MAB* as it receives all demonstrations immediately. However, *contextual MAB* converges to a policy with a high success rate more quickly. This indicates that *contextual MAB* received more informative demonstrations by asking for demonstrations at initial states where a good policy had not yet been learnt. *RL only* failed to learn a good policy in the limited number of episodes. Our results show that setting $\epsilon = 0$ for the Q-filter resulted in slower learning by filtering out most demonstration experience from the behaviour cloning loss. A promising approach may be to start with a large value for ϵ, and then gradually decay ϵ.

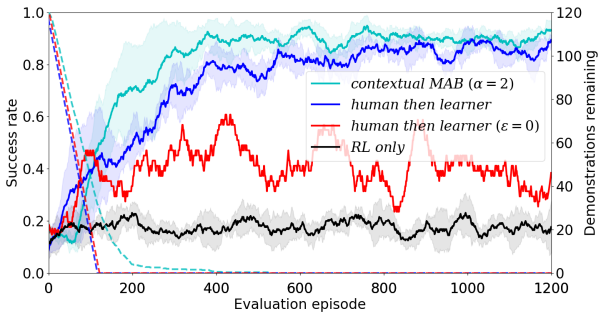


Fig. 4: Success rate on evaluation episodes for policies trained with a limited budget of 120 human demonstrations. Results are averaged over 5 runs for each method. Solid lines indicate average success rate over a sliding window of 40 episodes. Shaded regions illustrate the standard deviation over the 5 runs. Dashed lines indicate the demonstrations remaining.

G. Real-World Experiment

In the real-world experiment we compared *contextual MAB* and *human then learner* over one run each of 400 episodes. In the real-world experiments, we used $\epsilon = 0.1$ and a larger neural network with hidden layers of 128 and 64 weights as this was empirically found to improve the performance of the learnt policies. All other training parameters were the same as for the simulated experiments. For *contextual MAB* we used $\alpha = 1$. The costs were $c_d = 1$ and $c_f = 5$.

Results: *contextual MAB* used 142 human demonstrations and incurred a total cost of 442 over the 400 total episodes. With exactly the same number of demonstrations, *human then learner* incurred more failures, accumulating a total cost of 517. The cost incurred throughout the experiment is plotted in Figure 5. These results demonstrate that our approach can easily be applied to real-world robotics problems.

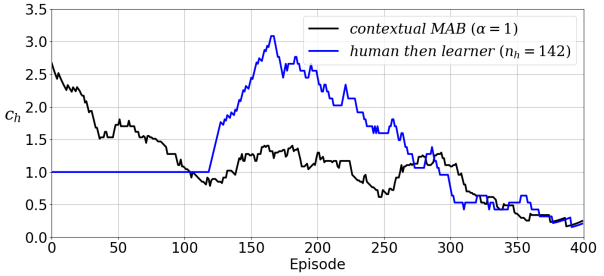


Fig. 5: Human cost in real-world navigation experiment. c_h is plotted using a sliding window average over 40 episodes.

H. Discussion

The α value in our approach enables the user to tune whether the system conservatively asks for more demonstrations, or explores trying the other controllers. A smaller value reduces the risk of failures, but with a higher value the system

more quickly gathers RL experience and more accurately predicts controller performance. Our results from our two simulated domains indicate that simple tasks that can be learnt from demonstrations alone favour a smaller value for α to conservatively ask for demonstrations and avoid failures. For more difficult problems which take longer to learn and require RL experience to train a good policy, a larger value may be favourable to gather more RL experience.

Choosing the controller for each episode based on context is key to the success of our approach. The *Boltzmann* method favours choosing the learnt policy when it begins to perform well overall. However, our method performs better by selectively favouring the autonomous controllers only from initial states where they are likely to succeed. As a side effect, our approach tends to ask for more informative demonstrations, resulting in better performance of the learnt policy with the same number of demonstrations.

Our results indicate that including demonstrations from different sources increases the amount of experience required to learn a good policy. It may be useful to use our approach to learn from many different controllers (for example, a number of different teleoperators who have different techniques). However, the issue of learning effectively from dissimilar demonstrations may need to be addressed before our approach is advantageous with a large number of different controllers.

VI. Conclusion

We have presented an approach to choosing between human teleoperation and autonomous controllers to minimise the cost of bothering the human operator. By estimating the performance of each controller throughout the state space, our system reduces the human cost by only asking for demonstrations when they are needed, and reducing autonomous failures. By learning one of the autonomous controllers online, our system becomes less reliant on the human with more experience. In future work, we will apply the framework presented in this paper to problems where the human operator may make mistakes, and so we will also need to predict human performance. Additionally, we will extend our approach to sequential decision making problems to consider *sequences* of controller selections. To further reduce the human effort required, our framework could be applied using more data-efficient learning methods for the learnt controller, such as model-based RL methods.

References

[1] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *AAAI Conference on Artificial Intelligence*, 2018.

[2] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Robotics: Science and Systems*, 2018.

[3] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *ICRA*, 2018.

[4] G. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost, "Adjustable autonomy for human-centered autonomous systems on mars," in *Mars society conference*, 1998.

[5] N. Hawes *et al.*, "The STRANDS project: Long-term autonomy in everyday environments," *IEEE RAM*, vol. 24, no. 3, 2017.

[6] M. Chiou, R. Stolkin, G. Bieksaite, N. Hawes, K. L. Shapiro, and T. S. Harrison, "Experimental analysis of a variable autonomy framework for controlling a remotely operating mobile robot," in *IROS*, 2016.

[7] K. H. Wray, L. Pineda, and S. Zilberstein, "Hierarchical approach to transfer of control in semi-autonomous systems," in *AAMAS*. IFAAMAS, 2016.

[8] N. Jansen, M. Cubuktepe, and U. Topcu, "Synthesis of shared control protocols with provable safety and performance guarantees," in *ACC*. IEEE, 2017.

[9] J. A. Clouse, *On integrating apprentice learning and reinforcement learning*. University of Massachusetts Amherst, 1996.

[10] F. Del Duchetto, A. Kucukyilmaz, L. Iocchi, and M. Hanheide, "Do not make the same mistakes again and again: Learning local recovery policies for navigation from human demonstrations," *IEEE RA-L*, vol. 3, no. 4, 2018.

[11] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *JAIR*, vol. 34, 2009.

[12] Z. Lin, B. Harrison, A. Keech, and M. O. Riedl, "Explore, exploit or listen: Combining human feedback and policy model to speed up deep reinforcement learning in 3d worlds," *arXiv preprint arXiv:1709.03969*, 2017.

[13] S. C. Akkaladevi, M. Plasch, C. Eitzinger, S. C. Maddukuri, and B. Rinner, "Towards learning to handle deviations using user preferences in a human robot collaboration scenario," in *International Conference on Intelligent Human Computer Interaction*. Springer, 2016, pp. 3–14.

[14] A. G. Barto and M. T. Rosenstein, "Supervised actor-critic reinforcement learning," *Handbook of learning and approximate dynamic programming*, vol. 2, 2004.

[15] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *Ro-Man*. IEEE, 2011.

[16] S. Reddy, A. D. Dragan, and S. Levine, "Shared autonomy via deep reinforcement learning," in *Robotics: Science and Systems XIV*, 2018.

[17] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *ICRA*, 2017.

[18] A. Broad, T. D. Murphey, and B. Argall, "Learning models for shared control of human-machine systems with unknown dynamics," *Robotics: Science and Systems XIII*, 2017.

[19] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *AAMAS*, 2013, pp. 1053–1060.

[20] J. Fu and U. Topcu, "Synthesis of shared autonomy policies with temporal logic specifications," *IEEE T-ASE*, vol. 13, no. 1, 2016.

[21] R. Cohen, H. Jung, M. W. Fleming, and M. Y. Cheng, "A user modeling approach for reasoning about interaction sensitive to bother and its application to hospital decision scenarios," *User Modeling and User-Adapted Interaction*, vol. 21, no. 4-5, pp. 441–484, 2011.

[22] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," *The International Journal of Robotics Research*, 2019.

[23] J. P. Fentanes, B. Lacerda, T. Krajník, N. Hawes, and M. Hanheide, "Now or later? Predicting and maximising success of navigation actions from long-term experience," in *ICRA*. IEEE, 2015.

[24] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *AAMAS*, 2006, pp. 720–727.

[25] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[26] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *WWW*. ACM, 2010.

[27] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *ALT*. Springer, 2011.

[28] R. Goetschalckx, P. Poupart, and J. Hoey, "Continuous correlated beta processes," in *IJCAI*, 2011.

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *ICLR*, 2016.

[30] H. Jakab and L. Csató, "Improving Gaussian process value function approximation in policy gradient algorithms," in *ICANN*, 2011.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IROS*, 2004.

[32] G. Echeverria, S. Lemaignan, A. Degroote, S. Lacroix, M. Karg, P. Koch, C. Lesire, and S. Stinckwich, "Simulating complex robotic scenarios with morse," in *SIMPAR*, 2012.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

8.1 Limitations and Future Work

There are several limitations to the approach that we proposed in this work. One is the simplistic approach that we took towards modelling the human in our framework. We assumed that the workload for the human can be modelled as fixed costs for resetting failures and for demonstrations. However, in reality the perceived workload for the human operator might depend on many factors. For example, periodically interrupting the human to request demonstrations may be more inconvenient than providing a large number of demonstrations upfront. Furthermore, requesting help from the human may be a greater burden if the human is currently busy. A more flexible approach that takes into account these additional human factors could be a challenging, but interesting direction for future work. Such research would likely require trials with real human operators to understand how the approach could be tailored towards human preferences.

The main limitation of the algorithm we proposed is that it is myopic in the sense that it determines what will minimise the cost based only on the current performance of the policy. It does not take into account the fact that the policy is able to improve, so providing a demonstration at the current episode will improve the policy in the long run, and prevent cost incurred due to future failures. Quantifying the improvement in the policy that can be expected by gathering additional demonstrations (as well as the associated reduction in future cost) appears to be a very difficult problem. However, this would be an interesting direction to try and improve upon the framework that we proposed.

A recent piece of work [182] tries to address this. The authors train a classifier to predict which tasks can be solved after receiving a demonstration for a given input task. This can then be used to weigh the long-term benefit of requesting additional demonstrations. However, this approach requires a large dataset of demonstrations across many tasks to train the classifier, which appears to undermine the original goal of reducing the human workload. A simpler alternative may be to include a reward bonus for receiving a demonstration to reflect the improvement in long-term cost. However, appropriately designing the reward bonuses would be very difficult.

A key attribute of our approach is the use of uncertainty estimation. We use uncertainty estimates to determine which tasks we are certain the robot will fail at, and therefore requires a demonstration. We also use the uncertainty estimates to determine which tasks we are unsure about the performance of the current policy, and therefore it might be worth attempting to use the policy. We used a simple kernel-based method for uncertainty estimation. In future work, our approach could be combined with uncertainty estimation techniques for powerful deep learning models to address more complex problems.

A final limitation of our approach is that it is only applicable to tasks with a binary outcome of success or failure. A direction for future work would be to develop an approach for tasks where the outcome for an episode is summarised by the cumulative reward received. This would lead to a multi-objective problem, where the human effort cost must be traded off against the reward received. Another direction would be to address problems where control may switch back and forth between human and demonstrator multiple times within an episode [24].

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	A framework for learning from demonstration with minimal human effort.
Publication Status	□Published ☑Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Marc Rigter, Bruno Lacerda, and Nick Hawes (2020). A framework for learning from demonstration with minimal human effort. IEEE Robotics and Automation Letters (RAL).

Student Confirmation

Student Name:	Marc Rigter		
Contribution to the Paper	The idea was developed jointly by Bruno Lacerda, Nick Hawes, and myself. I implemented the algorithm and ran the experiments. I led the writing in collaboration with Bruno Lacerda and Nick Hawes.		
Signature		Date	16th November 2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Nick Hawes		
Supervisor comments I confirm that Marc made a substantial contribution to the publication, and that the description above is accurate.			
Signature		Date	22nd November 2022

This completed form should be included in the thesis, at the end of the relevant chapter.

9 Conclusion

This thesis has contributed towards developing algorithms for sequential decision-making that are able to reckon with uncertainty in a number of different problem settings. In Chapter 3, we considered problems where the MDP is fully specified and we wish to optimise CVaR to be risk-averse to the stochastic outcomes of the MDP. We developed an approach to achieve the best possible expected performance whilst ensuring CVaR is optimal. In Chapters 4 and 5 we considered problems where there is uncertainty over the MDP specified by a set of MDPs, or a prior distribution over MDPs, respectively. In Chapter 4 we developed approximations to finding a policy with the lowest maximum sub-optimality, or regret, over the set of MDPs. In Chapter 5 we considered the optimisation of CVaR in the presence of the uncertain distribution over MDPs to mitigate risk due to both epistemic and aleatoric uncertainty. In Chapters 6 and 7, we proposed new algorithms for offline RL. Our approach in Chapter 6 is based on developing a scalable approach to solving a type of robust MDP. In Chapter 7, we proposed an approach that tackles the problem of offline RL by finding a policy that is risk-averse to both epistemic and aleatoric uncertainty, in a similar spirit to Chapter 5. Finally, in Chapter 8 we considered the learning from demonstration setting where we must reason about the uncertain performance of the learnt policy to minimise the human effort required to train the policy.

9.1 Discussion

In this thesis we have considered both tabular approaches, and approaches that utilise deep learning. There are clear tradeoffs between the two approaches. On the one hand, tabular approaches can often guarantee an exact solution, and that solution can be easily interpreted. However, these approaches cannot be scaled beyond small problems. On the other hand, deep RL is considerably more scalable, but has the downside that it can be sensitive to hyperparameters, may require abundant training data, and lacks guarantees on the performance of the final policy. It remains unclear whether there is a satisfactory middle ground to this tradeoff, but we think that conducting research under both paradigms is worthwhile. Addressing tabular problems is often a better way to understand the core elements of the problem at hand, and is a useful foundation from which to try and develop algorithms that are more scalable.

One of the core arguments that we have made throughout this thesis is that when designing sequential decision-making algorithms, we should take into consideration both epistemic and aleatoric uncertainty. Much research on both planning and RL disregards epistemic uncertainty by either assuming access to a perfect model, or to a perfect simulator from which unlimited experience can be obtained. However, for many real problems there is no perfect model or simulator, and we only have access to limited data from the environment. Therefore, there will always be inaccuracies in any model, simulator, or value function derived from this data. By explicitly reasoning about model uncertainty in the model-based approaches that we have considered, we can ensure that the policies we generate make sound decisions even in the presence of limited knowledge about the environment.

Furthermore, going beyond optimising for the expectation and considering risk-sensitive objectives is crucial for mitigating aleatoric risk in stochastic environments. Much of the current research in sequential decision-making, particularly in deep RL, focuses on deterministic benchmark domains. We think that more emphasis should be placed on evaluating the performance of algorithms, including their variability

in performance, in highly stochastic domains which are more representative of the real world.

Rather than considering each uncertainty source in isolation, like much of the previous work in this area, we think that more research should investigate approaches that jointly consider both epistemic and aleatoric uncertainty. After all, our goal should be to develop policies that avoid catastrophic outcomes, irrespective of the source of uncertainty that led to that outcome. Focussing on this goal may improve progress toward developing algorithms that can be deployed safely in the real world in stochastic and safety-critical environments with access to limited data.

9.2 Future Work

To close, we discuss some concrete research directions that lead on from the work in this thesis. In each chapter, we have already discussed the limitations of each individual piece of work, and suggested future work to address those limitations. In this final section, we focus on directions for future work that draw connections between multiple chapters.

Better Models An orthogonal direction to the research presented in this thesis is to develop better machine learning models for capturing uncertainty. Throughout this thesis, we have used methods for representing epistemic uncertainty over the MDP model that are either difficult to scale (discrete sets of MDPs, Dirichlet distributions), or may be unreliable (ensembles of neural networks). Furthermore, to capture aleatoric uncertainty, our neural network models used Gaussian distributions over successor states which may be highly inaccurate for some problems, particularly those with multi-modal dynamics. Therefore, developing machine learning models that are scalable, and can accurately model both epistemic and aleatoric uncertainty, is crucial to improving the practical effectiveness of the approaches presented in this work. Integrating recent advances in these areas [158] with the algorithms presented in this thesis is a straightforward next step for future work.

Adaptive Risk-Averse Offline RL In Chapter 5 we generated policies that are risk-averse to epistemic and aleatoric uncertainty, but adapt their behaviour as epistemic uncertainty is reduced during online execution. In Chapter 7 we addressed risk-aversion to epistemic and aleatoric uncertainty for offline RL, but trained a Markovian policy which did not allow for adaptivity. Throughout this thesis, we have argued that risk-averse algorithms are crucial for the safety-critical applications in which offline RL is most applicable. Meanwhile, concurrent works have argued for training adaptive policies for offline RL [183, 184], as this helps to overcome the limitations of a fixed dataset by enabling the policy to quickly adapt to the true environment.

Thus, a natural direction for future work is to combine the concepts of risk-sensitivity and adaptiveness in the context of offline RL, and seek to train risk-averse policies that are adaptive. Adaptivity might help enable the generation of more performant policies from very limited datasets, while incorporating risk-sensitivity might make these approaches more applicable to safety-critical environments.

Minimax Regret for Offline RL In Chapter 6, we developed an algorithm to solve a robust MDP formulation of offline RL, where we found the policy with the best worst-case expected value across a set of candidate MDPs. In Chapter 4, we addressed the minimax regret criterion for an uncertainty set of MDPs. The argument for considering the minimax regret criterion, as opposed to maximin expected value (i.e. robust MDPs), is that minimax regret is less conservative but can still be considered robust as it optimises for the worst-case sub-optimality.

A possible direction for future work is to address the minimax regret objective in the deep offline RL setting. This would involve developing a more scalable approach for optimising minimax regret, something that we have already discussed in Section 4.1. A potential advantage of this approach is that it might result in more aggressive and performant policies than existing approaches which are usually very pessimistic and consider either worst-case optimisation (such as Chapter 6), or other methods that lower-bound the value function in the true environment [140, 153].

By considering the worst-case regret across all plausible environments, one would expect that the resulting policy should still be robust to environment uncertainty.

Pareto Fronts for Risk-Sensitive Policies In Chapter 3 we explored the fact that optimising for risk alone may not achieve the best possible tradeoff with the expected value. In Chapter 3 we proposed to optimise for the best expected value, given that performance is optimal for a chosen risk-level. However, it is not clear how to choose the level of risk to optimise for that obtains the desired tradeoff between these two objectives.

Therefore, a possible future direction is to instead focus on developing scalable algorithms that can efficiently generate a Pareto front of policies that obtain different tradeoffs between risk and expected value. After observing the behaviours of the different policies, the desired tradeoff may be selected. This approach might additionally be useful in the offline RL setting, where it has been shown that generating a Pareto front of diverse policies with different levels of pessimism and selecting the best policy can be used to obtain very strong performance [157]. Generating and selecting between multiple candidate policies with different levels of risk-aversion may be a promising alternative.

Robustness and Risk-Sensitivity for Human-Robot Interaction In Chapter 8 we briefly touched upon human-robot interaction, in the form of interactively training a policy using demonstrations. In Chapter 8, we did not consider any form of risk-sensitivity or robustness, and furthermore we assumed that the human demonstrator never made any mistakes. In future work, we think that tasks involving human-robot interaction and collaboration are a natural domain in which to apply some of the ideas developed in this thesis.

There are a number of ways in which risk-sensitivity and/or robustness could be applied to this setting. One possible direction would be to consider robustness or risk-sensitivity towards human behaviour. Human behaviour is both unpredictable, and variable between humans, and therefore cannot easily be specified by a hand-crafted model [24]. Therefore, when developing algorithms for human robot-interaction,

the human should be modelled using data-driven approaches. Optimising for a risk-sensitive or robust objective in the presence of an uncertain human model that is derived from data, in the spirit of Chapters 6 and 7, may be an effective means to handle the high degree of unpredictability of humans in collaborative tasks.

Furthermore, adaptive policies like those obtained in Chapter 5 will be useful for adapting to the unique characteristics of each human, when interacting with multiple humans. Recent work [24] has begun to explore this direction.

Further Applications We wish to apply the ideas in this thesis to other real-world problems, aside from human-robot interaction. In robotics, some possibilities include applying the offline RL algorithms developed in this thesis to real systems [185], or using algorithms for robust or risk-sensitive policy optimisation to improve sim-to-real transfer [186]. Other possible application domains include finance and healthcare. In these domains, accurate simulators do not exist, and incorporating risk and uncertainty into decision-making is paramount.

The goal of this thesis is to make a small contribution towards the widespread application of planning and reinforcement learning algorithms to real world problems. Our hope is that if, and when, these algorithms become widely used, their benefits are distributed equitably.

Bibliography

[1] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[2] Huan-Hsin Tseng, Yi Luo, Sunan Cui, Jen-Tzung Chien, Randall K Ten Haken, and Issam El Naqa. Deep reinforcement learning for automated radiation adaptation in lung cancer. *Medical physics*, 44(12):6690–6705, 2017.

[3] Flora Charbonnier, Thomas Morstyn, and Malcolm D McCulloch. Scalable multi-agent reinforcement learning for distributed control of residential energy flexibility. *Applied Energy*, 314:118825, 2022.

[4] Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis. Market making via reinforcement learning. *International Conference on Autonomous Agents and Multiagent Systems*, 2018.

[5] Carl Benedikt Frey and Michael A Osborne. The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, 114:254–280, 2017.

[6] Anirudha Majumdar and Marco Pavone. How should a robot assess risk? Towards an axiomatic theory of risk in robotics. In *Robotics Research*, pages 75–84. Springer, 2020.

[7] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in Neural Information Processing Systems*, 30, 2017.

[8] Omar Khan, Pascal Poupart, and James Black. Minimal sufficient explanations for factored Markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 19, pages 194–200, 2009.

[9] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *International Conference on Learning Representations*, 2021.

[10] Shie Mannor, Duncan Simester, Peng Sun, and John N Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

[11] Frank Hyneman Knight. *Risk, uncertainty and profit*, volume 31. Houghton Mifflin, 1921.

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

[13] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[14] Shie Mannor, Ofir Mebel, and Huan Xu. Robust MDPs with k-rectangular uncertainty. *Mathematics of Operations Research*, 41(4):1484–1509, 2016.

[15] Nat Dilokthanakul and Murray Shanahan. Deep reinforcement learning with risk-seeking exploration. In *International Conference on Simulation of Adaptive Behavior*, pages 201–211. Springer, 2018.

[16] Marc Rigter, Bruno Lacerda, and Nick Hawes. A framework for learning from demonstration with minimal human effort. *IEEE Robotics and Automation Letters*, 5(2):2023–2030, 2020.

[17] Marc Rigter, Bruno Lacerda, and Nick Hawes. Minimax regret optimisation for robust planning in uncertain Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11930–11938, 2021.

[18] Marc Rigter, Bruno Lacerda, and Nick Hawes. Risk-averse Bayes-adaptive reinforcement learning. *Advances in Neural Information Processing Systems*, 34:1142–1154, 2021.

[19] Marc Rigter, Paul Duckworth, Bruno Lacerda, and Nick Hawes. Planning for risk-aversion and expected value in MDPs. *International Conference on Automated Planning and Scheduling*, 2022.

[20] Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: Robust adversarial model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

[21] Marc Rigter, Bruno Lacerda, and Nick Hawes. One risk to rule them all: A risk-sensitive perspective on model-based offline reinforcement learning. *arXiv preprint arXiv:2212.00124*, 2022.

[22] Shu Ishida, Marc Rigter, and Nick Hawes. Robot path planning for multiple target regions. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.

[23] Marc Rigter, Danial Dervovic, Parisa Hassanzadeh, Jason Long, Parisa Zehtabi, and Daniele Magazzeni. Optimal admission control for multiclass queues with time-varying arrival rates via state abstraction. *AAAI Conference on Artificial Intelligence*, 2022.

[24] Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. Shared autonomy systems with stochastic operator models. In *International Joint Conferences on Artificial Intelligence*, 2022.

[25] Anna Gautier, Marc Rigter, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. Risk-constrained planning for multi-agent systems with shared resources. *International Conference on Autonomous Agents and Multiagent Systems*, 2022.

[26] Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. Planning with hidden parameter polynomial MDPs. *AAAI Conference on Artificial Intelligence*, 2022.

[27] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[28] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[29] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[30] Andrey Kolobov et al. *Planning with Markov decision processes: An AI perspective*. Morgan & Claypool Publishers, 2012.

[31] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[32] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[33] W Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis)design for autonomous driving. *arXiv preprint arXiv:2104.13906*, 2021.

[34] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[35] Kevin M Smith and Margaret P Chapman. On exponential utility and conditional value-at-risk as risk-averse performance criteria. *arXiv preprint arXiv:2108.01771*, 2021.

[36] Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

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

[38] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *Conference on Robot Learning*, 2021.

[39] Shengpu Tang and Jenna Wiens. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Machine Learning for Healthcare Conference*, pages 2–35. PMLR, 2021.

[40] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[41] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.

[42] Ronald A Howard and James E Matheson. Risk-sensitive Markov decision processes. *Management Science*, 18(7):356–369, 1972.

[43] Sid Browne. Optimal investment policies for a firm with a random risk process: Exponential utility and minimizing the probability of ruin. *Mathematics of operations research*, 20(4):937–958, 1995.

[44] Susanne Emmer, Marie Kratz, and Dirk Tasche. What is the best risk measure in practice? A comparison of standard measures. *Journal of Risk*, 2015.

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

[46] Matthew J Sobel. The variance of discounted Markov decision processes. *Journal of Applied Probability*, 19(4):794–802, 1982.

[47] Dotan Di Castro, Aviv Tamar, and Shie Mannor. Policy gradients with variance related risk criteria. *International Conference on Machine Learning*, 2012.

[48] Robert Sollis. Value at risk: A critical overview. *Journal of Financial Regulation and Compliance*, 2009.

[49] Jon Danielsson, Bjørn N Jorgensen, Sarma Mandira, Gennady Samorodnitsky, and Casper G De Vries. Subadditivity re-examined: The case for value-at-risk. Technical report, Cornell University Operations Research and Industrial Engineering, 2005.

[50] Laurent Favre and José-Antonio Galeano. Mean-modified value-at-risk optimization with hedge funds. *The Journal of Alternative Investments*, 5(2):21–25, 2002.

[51] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[52] Amir Ahmadi-Javid. Entropic value-at-risk: A new coherent risk measure. *Journal of Optimization Theory and Applications*, 155(3):1105–1123, 2012.

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

[54] Shaun S Wang. A class of distortion operators for pricing financial and insurance risks. *Journal of Risk and Insurance*, pages 15–36, 2000.

[55] Yinlam Chow and Mohammad Ghavamzadeh. Algorithms for CVaR optimization in MDPs. In *Advances in Neural Information Processing Systems*, pages 3509–3517, 2014.

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

[57] Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. In *AAAI Conference on Artificial Intelligence*, page 2993–2999, 2015.

[58] Nicole Bäuerle and Jonathan Ott. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3):361–379, 2011.

[59] Basel Committee et al. Fundamental review of the trading book: A revised market risk framework. *Consultative Document, October*, 2013.

[60] Kang Boda and Jerzy A Filar. Time consistent dynamic risk measures. *Mathematical Methods of Operations Research*, 63(1):169–186, 2006.

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

[62] Christopher Gagne and Peter Dayan. Two steps to risk sensitivity. *Advances in Neural Information Processing Systems*, 34:22209–22220, 2021.

[63] Vivek Borkar and Rahul Jain. Risk-constrained Markov decision processes. In *49th IEEE Conference on Decision and Control (CDC)*, pages 2664–2669. IEEE, 2010.

[64] LA Prashanth. Policy gradients for CVaR-constrained MDPs. In *International Conference on Algorithmic Learning Theory*, pages 155–169. Springer, 2014.

[65] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

[66] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. Worst cases policy gradients. *Conference on Robot Learning*, 2020.

[67] Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Sequential decision making with coherent risk. *IEEE Transactions on Automatic Control*, 62(7):3323–3338, 2016.

[68] Ramtin Keramati, Christoph Dann, Alex Tamkin, and Emma Brunskill. Being optimistic to be conservative: Quickly learning a CVaR policy. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 4436–4443, 2020.

[69] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

[70] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Nonparametric return distribution approximation for reinforcement learning. In *International Conference on Machine Learning*, 2010.

[71] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

[72] Xiaoteng Ma, Li Xia, Zhengyuan Zhou, Jun Yang, and Qianchuan Zhao. DSAC: Distributional soft actor critic for risk-sensitive reinforcement learning. *arXiv preprint arXiv:2004.14547*, 2020.

[73] Núria Armengol Urpí, Sebastian Curi, and Andreas Krause. Risk-averse offline reinforcement learning. *International Conference on Learning Representations*, 2021.

[74] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88(3):411–424, 2000.

[75] Eric M Wolff, Ufuk Topcu, and Richard M Murray. Robust control of uncertain Markov decision processes with temporal logic specifications. In *IEEE Conference on Decision and Control*, pages 3372–3379. IEEE, 2012.

[76] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

[77] Aviv Tamar, Shie Mannor, and Huan Xu. Scaling up robust MDPs using function approximation. In *International Conference on Machine Learning*, pages 181–189. PMLR, 2014.

[78] Daniel J Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification. *International Conference on Learning Representations*, 2020.

[79] Erick Delage and Shie Mannor. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations research*, 58(1), 2010.

[80] Shie Mannor, Ofir Mebel, and Huan Xu. Lightning does not strike twice: Robust mdps with coupled uncertainty. In *International Conference on Machine Learning*, 2012.

[81] Yossiri Adulyasak, Pradeep Varakantham, Asrar Ahmed, and Patrick Jaillet. Solving uncertain MDPs with objectives that are separable over instantiations of model uncertainty. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[82] Asrar Ahmed, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Regret based robust solutions for uncertain Markov decision processes. In *Advances in Neural Information Processing Systems*, 2013.

[83] Asrar Ahmed, Pradeep Varakantham, Meghna Lowalekar, Yossiri Adulyasak, and Patrick Jaillet. Sampling based approaches for minimizing regret in uncertain Markov decision processes. *Journal of Artificial Intelligence Research*, 59, 2017.

[84] Katherine Chen and Michael Bowling. Tractable objectives for robust policy optimization. In *Advances in Neural Information Processing Systems*, pages 2069–2077, 2012.

[85] Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. Scenario-based verification of uncertain MDPs. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 287–305. Springer International Publishing, 2020.

[86] Lauren N Steimle, David L Kaufman, and Brian T Denton. Multi-model Markov decision processes. *Optimization Online*, 2018.

[87] Huan Xu and Shie Mannor. Parametric regret in uncertain Markov decision processes. In *Proceedings of the 48h IEEE Conference on Decision and Control*. IEEE, 2009.

[88] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[89] Alon Cohen, Haim Kaplan, Yishay Mansour, and Aviv Rosenberg. Near-optimal regret bounds for stochastic shortest path. *International Conference on Machine Learning*, 2020.

[90] Jean Tarbouriech, Evrard Garcelon, Michal Valko, Matteo Pirotta, and Alessandro Lazaric. No-regret exploration in goal-oriented reinforcement learning. *International Conference on Machine Learning*, 2020.

[91] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

[92] Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[93] Kevin Regan and Craig Boutilier. Robust online optimization of reward-uncertain MDPs. In *International Joint Conference on Artificial Intelligence*, 2011.

[94] Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under model mismatch. *Advances in Neural Information Processing Systems*, 30, 2017.

[95] Yue Wang and Shaofeng Zou. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34, 2021.

[96] Yufei Kuang, Miao Lu, Jie Wang, Qi Zhou, Bin Li, and Houqiang Li. 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, 2022.

[97] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 70, pages 2817–2826, 2017.

[98] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[99] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[100] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8522–8528. IEEE, 2019.

[101] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483, 2015.

[102] Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

[103] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[104] Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704, 2006.

[105] David Silver and Joel Veness. Monte-Carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010.

[106] Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. *Advances in Neural Information Processing Systems*, 25:1025–1033, 2012.

[107] Arthur Guez, Nicolas Heess, David Silver, and Peter Dayan. Bayes-adaptive simulation-based search with value function approximation. *Advances in Neural Information Processing Systems*, 27, 2014.

[108] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A very good method for Bayes-adaptive deep RL via meta-learning. *International Conference on Learning Representations*, 2020.

[109] Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013.

[110] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018.

[111] J Zico Kolter and Andrew Y Ng. Near-Bayesian exploration in polynomial time. In *International Conference on Machine Learning*, pages 513–520, 2009.

[112] Jonathan Sorg, Satinder Singh, and Richard L Lewis. Variance-based rewards for approximate Bayesian reinforcement learning. *Uncertainty in Artificial Intelligence*, 2010.

[113] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[114] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[115] Xinkun Nie, Emma Brunskill, and Stefan Wager. Learning when-to-treat policies. *Journal of the American Statistical Association*, 116(533):392–409, 2021.

[116] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: An empirical study. *Machine learning*, 84(1):109–136, 2011.

[117] Kirthevasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David Carter. Batch policy gradient methods for improving neural conversation models. *International Conference on Learning Representations*, 2017.

[118] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[119] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. IRIS: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420. IEEE, 2020.

[120] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[121] Chenjun Xiao, Ilbin Lee, Bo Dai, Dale Schuurmans, and Csaba Szepesvari. The curse of passive data collection in batch reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 8413–8438. PMLR, 2022.

[122] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *International Conference on Learning Representations*, 2021.

[123] Yu Bai, Tengyang Xie, Nan Jiang, and Yu-Xiang Wang. Provably efficient Q-learning with low switching cost. *Advances in Neural Information Processing Systems*, 32, 2019.

[124] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[125] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with Fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

[126] Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in Neural Information Processing Systems*, 34:27395–27407, 2021.

[127] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[128] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

[129] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21810–21823, 2020.

[130] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *International Conference on Learning Representations*, 2018.

[131] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

[132] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[133] Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[134] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[135] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

[136] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[137] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[138] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[139] Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

[140] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[141] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. *International Conference on Machine Learning*, 2022.

[142] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[143] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[144] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. *Advances in Neural Information Processing Systems*, 34:7436–7447, 2021.

[145] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[146] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021.

[147] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[148] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *International Conference on Machine Learning RL4RealLife Workshop*, 2019.

[149] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. AlgaeDICE: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[150] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[151] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

[152] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[153] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[154] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pages 619–629. PMLR, 2021.

[155] Catherine Cang, Aravind Rajeswaran, Pieter Abbeel, and Michael Laskin. Behavioral priors and dynamics models: Improving performance and domain transfer in offline RL. In *Deep RL Workshop NeurIPS 2021*, 2021.

[156] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104:104366, 2021.

[157] Yijun Yang, Jing Jiang, Tianyi Zhou, Jie Ma, and Yuhui Shi. Pareto policy pool for model-based offline reinforcement learning. In *International Conference on Learning Representations*, 2022.

[158] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana

Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[159] Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and Stephen J Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022.

[160] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

[161] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2020.

[162] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. In *International Conference on Machine Learning*, pages 7953–7963. PMLR, 2020.

[163] Toru Hishinuma and Kei Senda. Weighted model estimation for offline model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[164] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34, 2021.

[165] Stefan Schaal. Learning from demonstration. *Advances in Neural Information Processing Systems*, 9, 1996.

[166] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[167] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

[168] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pages 661–668, 2010.

[169] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 29, 2016.

[170] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[171] Jens Kober and Jan Peters. Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pages 2112–2118. IEEE, 2009.

[172] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International Conference on Machine Learning*, pages 3309–3318. PMLR, 2017.

[173] Jeffery Allen Clouse. *On integrating apprentice learning and reinforcement learning*. University of Massachusetts Amherst, 1996.

[174] Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34, 2009.

[175] Francesco Del Duchetto, Ayse Kucukyilmaz, Luca Iocchi, and Marc Hanheide. Do not make the same mistakes again and again: Learning local recovery policies for navigation from human demonstrations. *IEEE Robotics and Automation Letters*, 3(4), 2018.

[176] Marek Petrik and Dharmashankar Subramanian. An approximate solution method for large risk-averse Markov decision processes. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, page 805–814, 2012.

[177] John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J Kochenderfer. Bayesian optimized Monte Carlo planning. *AAAI Conference on Artificial Intelligence*, 2021.

[178] Finale Doshi-Velez and George Konidaris. Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *International Joint Conference on Artificial Intelligence*, 2016.

[179] Masatoshi Uehara and Wen Sun. Pessimistic model-based offline reinforcement learning under partial coverage. *International Conference on Learning Representations*, 2022.

[180] Hannes Eriksson, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis. Sentinel: taming uncertainty with ensemble based distributional reinforcement learning. In *Uncertainty in Artificial Intelligence*, pages 631–640. PMLR, 2022.

[181] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.

[182] Shivam Vats, Oliver Kroemer, and Maxim Likhachev. Synergistic scheduling of learning and allocation of tasks in human-robot teams. *International Conference on Robotics and Automation*, 2022.

[183] Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Advances in Neural Information Processing Systems*, 34:4607–4618, 2021.

[184] Dibya Ghosh, Anurag Ajay, Pulkit Agrawal, and Sergey Levine. Offline RL policies should be trained to be adaptive. In *International Conference on Machine Learning*, pages 7513–7530. PMLR, 2022.

[185] Nico Gürtler, Sebastian Blaes, Pavel Kolev, Felix Widmaier, Manuel Wuthrich, Stefan Bauer, Bernard Schölkopf, and Georg Martius. Benchmarking offline reinforcement learning on real-robot hardware. In *International Conference on Learning Representations*, 2023.

[186] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

Appendices of Chapter 3

Proof of Proposition 2

Let π denote a CVaR-optimal policy for confidence level α, and let σ denote the corresponding adversary policy from Equation 8. For some history, h, if $\mathcal{P}_\pi^{\mathcal{M}}(h) > 0$ and $\mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h) = 0$, there exists a policy $\pi' \in \Pi_{\mathcal{H}}^{\mathcal{M}}$ which may be executed from h onwards for which the total cost received over the run is guaranteed to be less than or equal to VaR$_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$.

Proof. We prove Proposition 2 by showing that the original policy π must satisfy this condition. Denote by $\mathcal{H}_g^{\mathcal{M}}$ the set all of histories ending at a goal state. We write $h_g \in \mathcal{H}_g^{\mathcal{M}}$ to denote a specific history ending at a goal state.

Consider a perturbation function, $\delta(h_g)$, such that $0 \leq \delta(h_g) \leq \frac{1}{\alpha}$ and $\sum_{h_g} \delta(h_g) \cdot \mathcal{P}_\pi^{\mathcal{M}}(h_g) = 1$. The expected value of $\mathcal{C}_\pi^{\mathcal{M}}$ under a distribution perturbed by δ is:

$$\mathbb{E}_\delta[\mathcal{C}_\pi^{\mathcal{M}}] = \sum_{h_g \in \mathcal{H}_g^{\mathcal{M}}} \delta(h_g) \cdot \mathcal{P}_\pi^{\mathcal{M}}(h_g) \cdot \text{cumul}_C(h_g). \quad (17)$$

By the CVaR representation theorem in Equation 2,

$$\max_\delta \mathbb{E}_\delta[\mathcal{C}_\pi^{\mathcal{M}}] = \text{CVaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}}). \quad (18)$$

Let $\mathcal{H}_{g,\text{VaR}}^{\mathcal{M}} \subseteq \mathcal{H}_g^{\mathcal{M}}$ denote the set of histories for which $\text{cumul}_C(h_g) = \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$. Any perturbation function which maximises Equation 18, denoted by δ^*, must satisfy the following conditions:

$$\delta^*(h_g) = 1/\alpha \quad \text{if} \quad \text{cumul}_C(h_g) > \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}}), \quad (19)$$

$$\delta^*(h_g) = 0 \quad \text{if} \quad \text{cumul}_C(h_g) < \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}}), \quad (20)$$

$$\sum_{h_g \in \mathcal{H}_{g,\text{VaR}}^{\mathcal{M}}} \delta^*(h_g) \cdot \mathcal{P}_\pi^{\mathcal{M}}(h_g) = 1 - \frac{1}{\alpha} \Pr(\mathcal{C}_\pi^{\mathcal{M}} > \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})), \quad (21)$$

where Equation 21 ensures that the perturbed probability distribution sums to 1. Notably, $\delta^*(h_g) = 0$ only if $\text{cumul}_C(h_g) \leq \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$.

From Proposition 1 we also have that:

$$\text{CVaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}}) = \sum_{h_g \in \mathcal{H}_g^{\mathcal{M}}} \mathcal{P}_{(\pi,\sigma)}^{\mathcal{M}}(h_g) \cdot \text{cumul}_C(h_g). \quad (22)$$

Therefore:

$$\sum_{h_g \in \mathcal{H}_g^{\mathcal{M}}} \delta^*(h_g) \cdot \mathcal{P}_\pi^{\mathcal{M}}(h_g) \cdot \text{cumul}_C(h_g) = \sum_{h_g \in \mathcal{H}_g^{\mathcal{M}}} \mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h_g) \cdot \text{cumul}_C(h_g) = \sum_{h_g \in \mathcal{H}_g^{\mathcal{M}}} \delta_\sigma(h_g) \cdot \mathcal{P}_\pi^{\mathcal{M}}(h_g) \cdot \text{cumul}_C(h_g) \quad (23)$$

where we define $\delta_\sigma(h_g) = \mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h_g)/\mathcal{P}_\pi^{\mathcal{M}}(h_g)$. The values of $\delta_\sigma(h_g)$ are constrained in the same manner as $\delta(h_g)$ by the definition of the CVaR SSPG. Therefore, Equation 23 implies that $\delta_\sigma(h_g)$ satisfies the same conditions as $\delta^*(h_g)$ in Equations 19-21. We have that $\mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h_g) = 0$ when $\mathcal{P}_\pi^{\mathcal{M}}(h_g) > 0$ only if $\delta_\sigma(h_g) = 0$. $\delta_\sigma(h_g) = 0$ only if $\text{cumul}_C(h_g) \leq \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$ by Equations 19-21. Therefore

$$\mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h_g) = 0 \text{ and } \mathcal{P}_\pi^{\mathcal{M}}(h_g) > 0 \implies \text{cumul}_C(h_g) \leq \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}}). \quad (24)$$

Consider some history, h, where $\mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h) = 0$ and $\mathcal{P}_\pi^{\mathcal{M}}(h) > 0$. For all histories $h_g \in \mathcal{H}_g^{\mathcal{M}}$ reachable after h under π (i.e. for which $\mathcal{P}_\pi^{\mathcal{M}}(h_g) > 0$), $\mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h_g) = 0$. Therefore, by Equation 24 all histories h_g reachable after h under π are guaranteed to have $\text{cumul}_C(h_g) \leq \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$. This completes the proof that for any history h, where $\mathcal{P}_{(\pi,\sigma)}^{\mathcal{G}}(h) = 0$ and $\mathcal{P}_\pi^{\mathcal{M}}(h) > 0$, by continuing to execute policy π, the total cost is guaranteed to be less than or equal to $\text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$. □

Proof of Proposition 3

During execution of policy π optimal for $\text{CVaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$, we may switch to any policy π' and still attain the same CVaR, provided that π' is guaranteed to incur total cost of less than or equal to $\text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$.

Proof. Let $\mathcal{H}_{g,\leq\text{VaR}}^{\mathcal{M}}$ denote the set of histories where $\text{cumul}_C(h_g) \leq \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$, and $\mathcal{H}_{g,>\text{VaR}}^{\mathcal{M}}$ denote the set of histories where $\text{cumul}_C(h_g) > \text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$. If for any $h_g \in \mathcal{H}_{g,>\text{VaR}}^{\mathcal{M}}$, there exists a policy $\pi' \neq \pi$ such that executing π' partway through h_g is guaranteed to receive total cost less than or equal to $\text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$, then switching to π' would improve the CVaR, contradicting that π is a CVaR-optimal policy. Therefore, there will never be a suitable $\pi' \neq \pi$ to switch to along any $h_g \in \mathcal{H}_{g,>\text{VaR}}^{\mathcal{M}}$.

If partway through any $h_g \in \mathcal{H}_{g,\leq\text{VaR}}^{\mathcal{M}}$, we switch to a valid π' instead of π, this can result in no total costs above the VaR by the definition of π' in Proposition 3. Therefore, switching to π' cannot modify the distribution of total costs which exceed $\text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$. Additionally, because no total costs which are less than or equal to $\text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$ can be increased above $\text{VaR}_\alpha(\mathcal{C}_\pi^{\mathcal{M}})$ by switching to π', the VaR cannot increase.

Because switching from π to π' cannot change the distribution of total costs above the VaR, and it cannot increase the VaR, we observe from the definition of CVaR in Equation 1 that the CVaR cannot be increased. Because π already attains the optimal CVaR, the strategy of switching to π' must attain the optimal CVaR. □

Histograms of Total Costs

In this section, we present histograms showing the total cost received over a number of runs for each method.

In Figure 5 we present the histograms for the Betting Game domain. We see that for $\alpha = 0.2$, both *CVaR-EV* and *CVaR-WC* have equivalent right tails of the distribution, resulting in equal $\text{CVaR}_{0.2}$. However, *CVaR-EV* has more of the distribution shifted towards the left, resulting in better expected cost. *EV* performs well in expectation, but there are many runs which obtain the maximum cost, resulting in poor CVaR performance.

For the Betting Game domain, we do not include histograms for optimising CVaR $\alpha = 0.02$, as the optimal policy is never to bet and the cost received is therefore always 95 (see Table 1).

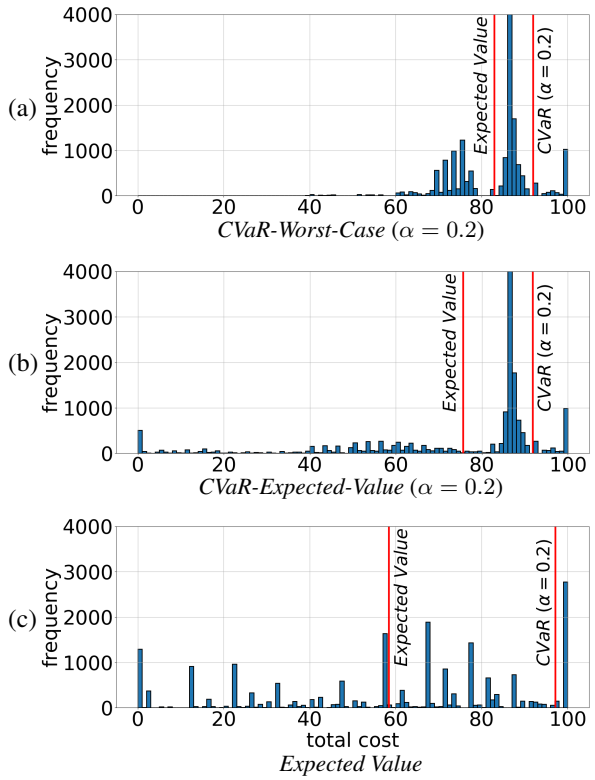


Figure 5: Histograms for the total cost received over 20000 evaluation episodes in the Betting Game domain.

Figure 6 shows the cost distributions for the Deep Sea Treasure domain. For $\alpha = 0.02$, we see that both *CVaR-EV* and *CVaR-WC* have equivalent distributions in the right most 2% of the distribution, resulting in equal $\text{CVaR}_{0.02}$. However, *CVaR-EV* has the total cost of more runs shifted to the left, resulting in lower expected cost. The same observations are made when optimising with $\alpha = 0.2$, but now the right most 20% of the distributions are equivalent, resulting in equal $\text{CVaR}_{0.2}$. For *EV*, a significant proportion of the runs obtain the maximum possible cost, resulting in poor CVaR performance.

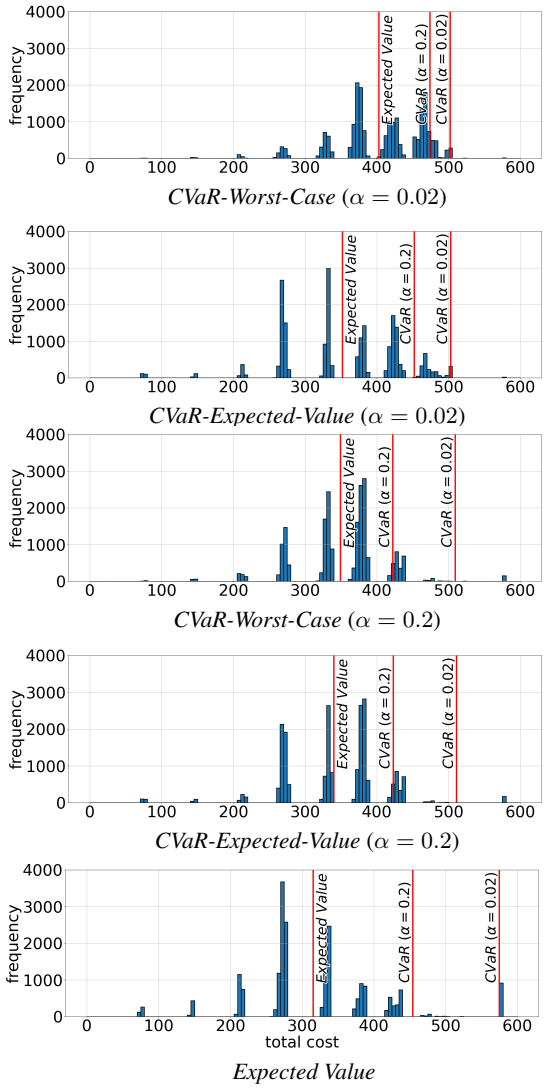


Figure 6: Histograms for the total cost received over 20000 evaluation episodes in the Deep Sea Treasure domain.

The cost distributions for the Autonomous Navigation domain are plotted in Figure 7. The same observations can be made for this domain as for the other domains. However, the difference between the distributions for *CVaR-EV* and *CVaR-WC* are more modest in the Autonomous Navigation domain.

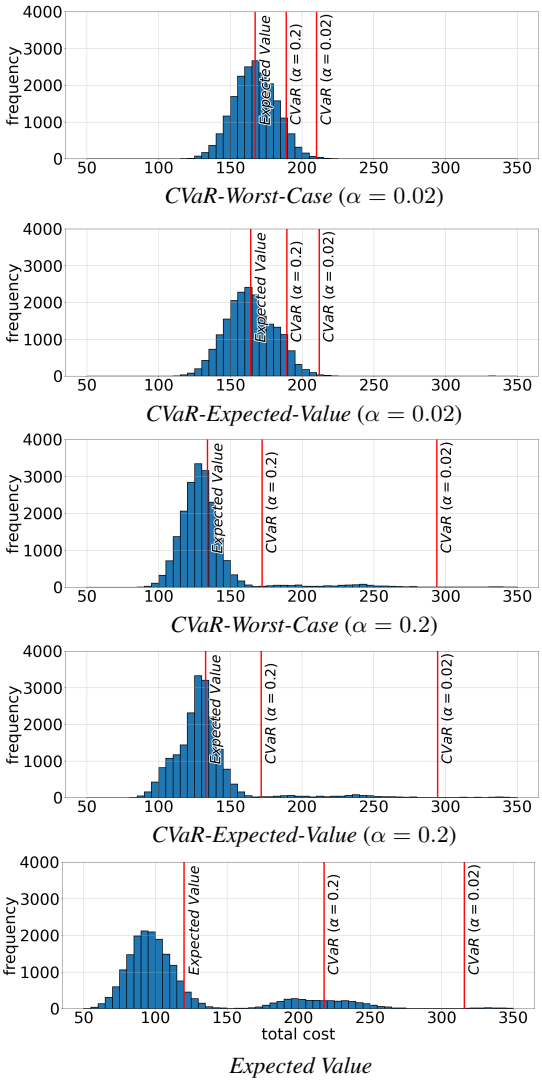


Figure 7: Histograms for the total cost received over 20000 evaluation episodes in the Autonomous Navigation domain.

Illustration of Deep Sea Treasure Domain

Figure 8 illustrates the Deep Sea Treasure domain.

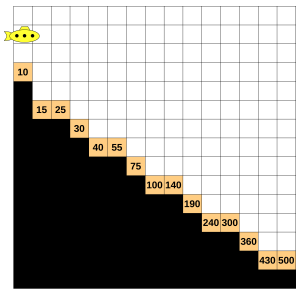


Figure 8: Deep Sea Treasure domain: golden squares indicate different treasure reward values.

Appendices of Chapter 4

Corrigendum

Proposition 2 of the paper is incorrect. The way that the adversary, σ^1, maximises regret in our approach is not equivalent to the maximising regret over independent uncertainty sets. As a result, our method is approximate for independent uncertainty sets.

Explanation

The way that our algorithm computes the maximum regret against the adversary, $\max_{\sigma^n} reg(s_0, \pi^n)$, means that for $n = 1$, the maximum regret computed by our algorithm is not equal to the maximum regret over independent uncertainty sets, i.e. $\max_{\sigma^1} reg(s_0, \pi) \neq \max_{\xi_q \in \xi} reg_q(s, \pi)$ for independent uncertainty set ξ.

Our approach in Algorithm 1 solves Problem 3 by finding the policy which minimises $\max_{\sigma^n} reg(s_0, \pi^n)$, where this quantity is computed according to:

$$\max_{\sigma^n} reg(s_0, \pi^n) := \max_{\xi_q \in \xi} \left\{ \sum_{o \in O} \pi^n(s, o) \cdot \left[V_q^n(s, \pi^o) + \sum_{s' \in S} T_q(s, o, s') \cdot V_q(s', \pi^*) - V_q(s, \pi^*) + \max_{\xi_{q'} \in \xi} \sum_{s' \in S} T_q(s, o, s') \cdot reg_{q'}(s', \pi^n) \right] \right\}. \quad (24)$$

For the case of $n = 1$ we find the policy which minimises $\max_{\sigma^1} reg(s_0, \pi)$ (i.e. Problem 2) which is computed according to:

$$\max_{\sigma^1} reg(s_0, \pi) := \max_{\xi_q \in \xi} \left\{ \sum_{a \in A} \pi(s, a) \cdot \left[\bar{C}_q(s, a) + \sum_{s' \in S} T_q(s, a, s') \cdot V_q(s', \pi_q^*) - \bar{C}_q(s, \pi_q^*(s)) - \sum_{s' \in S} T_q(s, \pi_q^*(s), s') \cdot V_q(s', \pi_q^*) + \max_{\xi_{q'} \in \xi} \sum_{s' \in S} T_q(s, a, s') \cdot reg_{q'}(s', \pi) \right] \right\} \quad (25)$$

To illustrate the discrepancy, we compare what is computed by our algorithm in Equation 25 to what is derived for the minimax regret using the Bellman equation in Proposition 1. From Proposition 1, the maximum regret over all uncertainty samples, $\xi_q \in \xi$, can be written as

$$\begin{aligned} \max_{\xi_q \in \xi} reg_q(s, \pi) &= \max_{\xi_q \in \xi} \left\{ \sum_{a \in A} \pi(s, a) \cdot \left[Q_q^{gap}(s, a) + \sum_{s' \in S} T_q(s, a, s') \cdot reg_q(s', \pi) \right] \right\} \\ &= \max_{\xi_q \in \xi} \left\{ \sum_{a \in A} \pi(s, a) \cdot \left[\bar{C}_q(s, a) + \sum_{s' \in S} T_q(s, a, s') \cdot V_q(s', \pi_q^*) - \bar{C}_q(s, \pi_q^*(s)) - \sum_{s' \in S} T_q(s, \pi_q^*(s), s') \cdot V_q(s', \pi_q^*) + \sum_{s' \in S} T_q(s, a, s') \cdot reg_q(s', \pi) \right] \right\} \end{aligned}$$

For independent uncertainty sets we can separately maximise over the member of the uncertainty set to applied at the current step, ξ_q, and that to be applied at all subsequent steps, $\xi_{q'}$:

$$\max_{\xi_q \in \xi} reg_q(s, \pi) = \max_{\xi_q, \xi_{q'} \in \xi} \left\{ \sum_{a \in A} \pi(s, a) \cdot \left[\bar{C}_q(s, a) + \sum_{s' \in S} T_q(s, a, s') \cdot V_{q'}(s', \pi_{q'}^*) - \bar{C}_q(s, \pi_q^*(s)) - \sum_{s' \in S} T_q(s, \pi_q^*(s), s') \cdot V_{q'}(s', \pi_{q'}^*) + \sum_{s' \in S} T_q(s, a, s') \cdot reg_{q'}(s', \pi) \right] \right\} \quad (26)$$

Crucially, the value function terms in Equation 26, $V_{q'}$, are computed under the member of the uncertainty set to be applied at all subsequent steps, ξ_q'. However, our approach in Equation 25 instead utilises V_q. Therefore, intuitively we can think of our approach as optimising the regret accumulated by assuming that the optimal value function at successor states is computed using the worst-case sample for the current state.

As a result of this discrepancy, Proposition 2 is incorrect and the objective that our approach optimises for independent uncertainty sets is not equal to the maximum regret. Therefore, our algorithm does not provide an exact solution for this class of problems. Theorem 6 of (Ghavamzadeh, Petrik, and Chow 2016) proves that finding the maximum regret of a policy is NP-hard even for independent uncertainty sets, so we cannot expect to find a scalable approach to this problem.

Impact on the Remainder of the Paper

Propositions 3, 4, and 5 analyse the result of optimising the quantities computed by Equations 24 and 25 for problems with *dependent* uncertainty sets. The above error only concerns problems with independent uncertainty sets, so does not effect Propositions 3, 4, and 5. Furthermore, the above error is irrelevant to our experimental results which use domains with dependent uncertainties.

Appendices

Proof of Proposition 1

Proposition 1. *(Regret Bellman Equation) The regret for a proper policy, π, can be computed via the following recursion*

$$reg(s,\pi) = \sum_{a\in A} \pi(s,a)\cdot\big[Q^{gap}(s,a) + \sum_{s'\in S} T(s,a,s')\cdot reg(s',\pi)\big], \text{ where}$$

$$Q^{gap}(s,a) = \big[\bar{C}(s,a) + \sum_{s'\in S} T(s,a,s')\cdot V(s',\pi^*)\big] - V(s,\pi^*),$$

and $reg(s,\pi) = 0,\ \forall s \in G$.

To prove the proposition we show that if we apply the equations in Proposition 1 starting from the initial state we recover the definition of the regret for a policy given by Definition 2. We start by combining the equations stated in the proposition for the initial state

$$reg(s_0,\pi) = \sum_{a\in A} \pi(s_0,a)\cdot\left[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot V(s',\pi^*) - V(s_0,\pi^*) + \sum_{s'\in S} T(s_0,a,s')\cdot reg(s',\pi)\right]. \quad (27)$$

We can move $V(s_0,\pi^*)$ outside of the sum as it does not depend on a. We start unrolling the definition by substituting for $reg(s',\pi)$

$$reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A} \pi(s_0,a)\cdot\left[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot V(s',\pi^*) + \right.$$
$$\left.\sum_{s'\in S} T(s_0,a,s')\cdot\left[\sum_{a\in A}\pi(s',a)\cdot\left[\bar{C}(s',a) + \sum_{s''\in S} T(s',a,s'')\cdot V(s'',\pi^*) - V(s',\pi^*) + \sum_{s''\in S} T(s',a,s'')\cdot reg(s'',\pi)\right]\right]\right]. \quad (28)$$

Again, we can move $V(s',\pi^*)$ outside of the inner sum as it does not depend on a. Thus, Equation 28 can be rewritten as

$$reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A} \pi(s_0,a)\cdot\left[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot V(s',\pi^*) + \right.$$
$$\left.\sum_{s'\in S} T(s_0,a,s')\cdot\left[-V(s',\pi^*) + \sum_{a\in A}\pi(s',a)\cdot\left[\bar{C}(s',a) + \sum_{s''\in S} T(s',a,s'')\cdot V(s'',\pi^*) + \sum_{s''\in S} T(s',a,s'')\cdot reg(s'',\pi)\right]\right]\right]. \quad (29)$$

Cancelling terms, we have

$$reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A} \pi(s_0,a)\cdot\left[\bar{C}(s_0,a) + \right.$$
$$\left.\sum_{s'\in S} T(s_0,a,s')\cdot\left[\sum_{a\in A}\pi(s',a)\cdot\left[\bar{C}(s',a) + \sum_{s''\in S} T(s',a,s'')\cdot V(s'',\pi^*) + \sum_{s''\in S} T(s',a,s'')\cdot reg(s'',\pi)\right]\right]\right]. \quad (30)$$

After repeating the above process of unrolling the expression and cancelling terms for h steps we arrive at the following expression

$$reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A} \pi(s_0,a)\cdot\left[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot\left[\sum_{a\in A}\pi(s',a)\cdot\left[\bar{C}(s',a) + \dots\right.\right.\right.$$
$$\left.\left.\left.\sum_{s^{h\text{-}1}\in S} T(s^{h\text{-}2},a,s^{h\text{-}1})\cdot\left[\sum_{a\in A}\pi(s^{h\text{-}1},a)\cdot\left[\bar{C}(s^{h\text{-}1},a) + \sum_{s^h\in S} T(s^{h\text{-}1},a,s^h)\cdot V(s^h,\pi^*) + \sum_{s^h\in S} T(s^{h\text{-}1},a,s^h)\cdot reg(s^h,\pi)\right]\right]\dots\right]\right]\right]. \quad (31)$$

Taking $h \to \infty$ we have $s^h \in G$ under the definition of a proper policy. Thus, $V(s^h, \pi^*) = 0$ by the definition of goal states in an SSP MDP (Definition 1), and $reg(s^h, \pi) = 0$ by the definition of the regret decomposition in Proposition 1. This allows us to further simplify the expression to the following

$$reg(s_0, \pi) = -V(s_0, \pi^*) + \sum_{a \in A} \pi(s_0, a) \cdot \Bigg[\bar{C}(s_0, a) + \sum_{s' \in S} T(s_0, a, s') \cdot \Bigg[\sum_{a \in A} \pi(s', a) \cdot \Bigg[\bar{C}(s', a) + \dots \\ \sum_{s^{h-1} \in S} T(s^{h-2}, a, s^{h-1}) \cdot \Bigg[\sum_{a \in A} \pi(s^{h\text{-}1}, a) \cdot \bar{C}(s^{h\text{-}1}, a) \Bigg] \dots \Bigg] \Bigg] \Bigg]. \quad (32)$$

The nested sum is simply the expected cost of the policy, $V(s_0, \pi)$. Thus, the expression can be further simplified to give the final result $reg(s_0, \pi) = V(s_0, \pi) - V(s_0, \pi^*)$, which is the original definition for the regret of a policy. □

Proof of Proposition 2

Proposition 2. *If uncertainties are independent per Def. 5 then Problem 1 is equivalent to Problem 2.*

In Problem 1, the agent first chooses a policy. The adversary observes the policy of the agent and reacts by choosing the uncertainty sample to be applied to maximise the regret for the policy of the agent. In Problem 2, the adversary reacts to the policy of the agent by choosing the mapping from state-action pairs to uncertainty samples which maximises the regret for the policy of the agent. To prove the proposition, we show that in the case of independent uncertainty sets these adversaries are equivalent.

Let ξ_{ind} be an independent uncertainty set. Then each sample of model uncertainty is $\xi_q = (C_q \in \mathcal{C}, T_q \in \mathcal{T}) \in \xi_{ind}$. By Definition 5, $\mathcal{T} = \times_{(s,a) \in S \times A} \mathcal{T}^{s,a}$ where $\mathcal{T}^{s,a}$ is the set of possible distributions over S after applying a in s, and $\mathcal{C} = \times_{(s,a) \in S \times A} \mathcal{C}^{s,a}$ where $\mathcal{C}^{s,a}$ is the set of possible expected costs of applying a in s. We write $\times$ to denote the Cartesian product of sets.

The adversary in Problem 2, $\sigma^1 : S \times A \times \xi \to \{0, 1\}$, maps each state and action chosen by the agent to an MDP sample such that the regret for the policy of the agent is maximised. This means that at a state-action pair, s, a, the adversary may apply any sample, $\xi_q^{s,a} = (C_q^{s,a} \in \mathcal{C}^{s,a}, T_q^{s,a} \in \mathcal{T}^{s,a})$, where we write $\xi_q^{s,a}$ to denote the MDP sample applied at s, a. At another state-action pair s', a', the adversary may also apply any sample $\xi_q^{s',a'} = (C_q^{s',a'} \in \mathcal{C}^{s',a'}, T_q^{s',a'} \in \mathcal{T}^{s',a'})$, and so on. Thus, over all state-action pairs, the adversary may choose any combination of different samples at each state-action pair. We can write the set of all possible combinations of transition and cost functions as $\times_{(s,a) \in S \times A} \mathcal{T}^{s,a}$ and $\times_{(s,a) \in S \times A} \mathcal{C}^{s,a}$ respectively. These sets are equal to $\mathcal{T}$ and $\mathcal{C}$ respectively by the definition of independence. Thus, over all state-action pairs, the combination of samples chosen by σ^1 is equivalent to $\xi_q = (C_q \in \times_{(s,a) \in S \times A} \mathcal{C}^{s,a}, T_q \in \times_{(s,a) \in S \times A} \mathcal{T}^{s,a}) = (C_q \in \mathcal{C}, T_q \in \mathcal{T}) \in \xi_{ind}$, such that the regret for the policy of the agent is maximised. The adversary in Problem 1 also chooses any $\xi_q = (C_q \in \mathcal{C}, T_q \in \mathcal{T}) \in \xi_{ind}$ such that the regret for the agent is maximised. Thus, we observe for independent uncertainties, the two adversaries maximising the regret are equivalent. Therefore $\operatorname{argmin}_{\pi \in \Pi} \max_{\xi_q \in \xi} reg_q(s_0, \pi) = \operatorname{argmin}_{\pi \in \Pi} \max_{\sigma^1} reg(s_0, \pi)$. □

Proof of Proposition 3

Proposition 3. *If the expected number of steps for π to reach $s_g \in G$ is at most H for any adversary:*

$$0 \leq \max_{\sigma^1} reg(s_0, \pi) - \max_{\xi_q \in \xi} reg_q(s_0, \pi) \leq (\delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H)H, \text{ where}$$

$$\begin{aligned} |\bar{C}_i(s, a) - \bar{C}_j(s, a)| &\leq \delta_C && \forall s \in S, a \in A, \xi_i \in \xi, \xi_j \in \xi \\ \textstyle\sum_{s'} |T_i(s, a, s') - T_j(s, a, s')| &\leq 2\delta_T && \forall s \in S, a \in A, \xi_i \in \xi, \xi_j \in \xi \\ |V_i(s, \pi^*) - V_j(s, \pi^*)| &\leq \delta_{V^*} && \forall s \in S, \xi_i \in \xi, \xi_j \in \xi \\ \bar{C}_i(s, a) &\leq C_{max} && \forall s \in S, a \in A, \xi_i \in \xi \end{aligned}$$

For the lower bound, we observe that the adversary σ^1 can apply different combinations of $\xi_q \in \xi$ at each step, and therefore is more powerful than an adversary that chooses only a single $\xi_q \in \xi$. Therefore we have

$$\max_{\sigma^1} reg(s_0, \pi) \geq \max_{\xi_q \in \xi} reg_q(s_0, \pi) \implies \max_{\sigma^1} reg(s_0, \pi) - \max_{\xi_q \in \xi} reg_q(\pi) \geq 0. \quad (33)$$

To prove the upper bound, we begin by denoting the most adversarial uncertainty sample by $\xi_\top = (C_\top, T_\top) = \operatorname{argmax}_{\xi_q \in \xi} reg_q(s_0, \pi)$. The corresponding value of regret for the most adversarial sample is denoted $reg_\top(s, \pi)$. We introduce the following expression for the error in the maximum regret value at a given state

$$f(s) = \max_{\sigma^1} reg(s, \pi) - reg_\top(s, \pi). \quad (34)$$

Thus we wish to find an upper bound for $f(s_0)$. We begin by expanding the expression for $f(s)$ using the regret decomposition from Proposition 1

$$f(s) = \max_{\sigma^1} reg(s,\pi) - reg_\top(s,\pi)$$
$$= \max_{\xi_q \in \xi} \Bigg[\sum_{a\in A} \pi(s,a) \cdot \Bigg[\bar{C}_q(s,a) + \sum_{s'\in S} T_q(s,a,s') \cdot V_q(s',\pi^*) - V_q(s,\pi^*) + \sum_{s'\in S} T_q(s,a,s') \cdot \max_{\sigma^1} reg(s',\pi) \Bigg] \Bigg] - \Bigg[\sum_{a\in A} \pi(s,a) \cdot \Bigg[\bar{C}_\top(s,a) + \sum_{s'\in S} T_\top(s,a,s') \cdot V_\top(s',\pi^*) - V_\top(s,\pi^*) + \sum_{s'\in S} T_\top(s,a,s') \cdot reg_\top(s',\pi) \Bigg] \Bigg]. \quad (35)$$

We can write the two max operators separately because when $n = 1$, the adversary can choose any $\xi_q \in \xi$ at each stage. We introduce the following notation for the difference between the true worst-case sample and another sample

$$\Delta T_q(s,a,s') = T_q(s,a,s') - T_\top(s,a,s'),$$
$$\Delta V_q(s,\pi^*) = V_q(s,\pi^*) - V_\top(s,\pi^*).$$

We also make a substitution for $\max_{\sigma^1} reg(s',\pi)$ using the definition of $f(s')$ from Equation 34. This results in the following expression

$$f(s) = \max_{\xi_q \in \xi} \Bigg[\sum_{a\in A} \pi(s,a) \cdot \Bigg[\bar{C}_q(s,a) + \sum_{s'\in S} (T_\top(s,a,s') + \Delta T_q(s,a,s')) \cdot (V_\top(s',\pi^*) + \Delta V_q(s',\pi^*)) - V_\top(s,\pi^*) - \Delta V_q(s,\pi^*) + \sum_{s'\in S} (T_\top(s,a,s') + \Delta T_q(s,a,s')) \cdot (reg_\top(s',\pi) + f(s')) \Bigg] \Bigg] -$$
$$\Bigg[\sum_{a\in A} \pi(s,a) \cdot \Bigg[\bar{C}_\top(s,a) + \sum_{s'\in S} T_\top(s,a,s') \cdot V_\top(s',\pi^*) - V_\top(s,\pi^*) + \sum_{s'\in S} T_\top(s,a,s') \cdot reg_\top(s',\pi) \Bigg] \Bigg]. \quad (36)$$

Cancelling terms and expanding, we have

$$f(s) = \max_{\xi_q \in \xi} \Bigg[\sum_{a\in A} \pi(s,a) \cdot \Bigg[\bar{C}_q(s,a) + \sum_{s'\in S} T_q(s,a,s') \cdot \Delta V_q(s',\pi^*) + \sum_{s'\in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) - \Delta V_q(s,\pi^*) + \sum_{s'\in S} T_q(s,a,s') \cdot f(s') + \sum_{s'\in S} \Delta T_q(s,a,s') \cdot reg_\top(s',\pi) \Bigg] \Bigg] - \sum_{a\in A} \pi(s,a) \cdot \bar{C}_\top(s,a). \quad (37)$$

At this point, we start to upper bound the terms in the previous equation using the constants defined in Proposition 3:

$$f(s) \leq \delta_C + 2\delta_{V^*} + \max_{\xi_q \in \xi} \Bigg[\sum_{a\in A} \pi(s,a) \Bigg[\sum_{s'\in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) + \sum_{s'\in S} T_q(s,a,s') \cdot f(s') + \sum_{s'\in S} \Delta T_q(s,a,s') \cdot reg_\top(s',\pi) \Bigg] \Bigg]. \quad (38)$$

We observe that

$$\sum_{s'\in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) \leq \delta_T \max_{s'\in S} V_\top(s',\pi^*) - \delta_T \min_{s'\in S} V_\top(s',\pi^*). \quad (39)$$

Under the assumption that for policy π the expected number of steps to reach the goal is at most H for any adversary, the it holds that

$$0 \leq V_\top(s',\pi^*) \leq V_\top(s',\pi) \leq C_{max} H. \quad (40)$$

Therefore

$$\sum_{s'\in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) \leq \delta_T C_{max} H. \quad (41)$$

By definition, $reg_\top(s,\pi) = V_\top(s,\pi) - V_\top(s,\pi^*)$, and can therefore similarly be bounded

$$0 \leq reg_\top(s',\pi) \leq C_{max}H, \tag{42}$$

$$\sum_{s'\in S} \Delta T_q(s,a,s') \cdot reg_\top(s',\pi^*) \leq \delta_T C_{max} H. \tag{43}$$

Combining with Equation 38 gives

$$f(s) \leq \delta_C + 2\delta_{V^*} + 2\delta_T C_{max}H + \max_{\xi_q\in\xi}\left[\sum_{a\in A}\pi(s,a)\cdot\left[\sum_{s'\in S} T_q(s,a,s')\cdot f(s')\right]\right]. \tag{44}$$

Thus, we can write for the initial state

$$\begin{aligned} f(s_0) \leq \delta_C + 2\delta_{V^*} + 2\delta_T C_{max}H + \\ \max_{\xi_q\in\xi}\left[\sum_{a\in A}\pi(s_0,a)\cdot\left[\sum_{s'\in S\setminus G} T_q(s_0,a,s')\cdot f(s') + \sum_{s'_g\in G} T_q(s_0,a,s'_g)\cdot f(s'_g)\right]\right] \\ \leq \Pr^{\pi,\sigma^1}(\tau^G_{s_0}=1)\cdot(\delta_C + 2\delta_{V^*} + 2\delta_T C_{max}H) + \\ \Pr^{\pi,\sigma^1}(\tau^G_{s_0}\geq 2)\cdot\left[\delta_C + 2\delta_{V^*} + 2\delta_T C_{max}H + \max_{s'\in S} f(s')\right], \end{aligned} \tag{45}$$

where in the last inequality we introduce the notation $\Pr^{\pi,\sigma^1}(\tau^G_{s_0}=h)$ to denote the probability that, under π and σ^1, a path starting from s_0 reaches a goal state in exactly h steps, and observe that $f(s'_g)=0$ for all $s'_g \in G$.

Using Equation 44 to substitute for $\max_{s'\in S} f(s')$ and repeatedly applying the same reasoning we have

$$f(s_0) \leq (\delta_C + 2\delta_{V^*} + 2\delta_T C_{max}H)\sum_{h=1}^{\infty}\Pr^{\pi,\sigma^1}(\tau^G_{s_0}=h)\cdot h \tag{46}$$

The sum on the right hand side is simply the expected number of steps to reach the goal and therefore we have the result

$$f(s_0) \leq (\delta_C + 2\delta_{V^*} + 2\delta_T C_{max}H)H. \quad \square \tag{47}$$

Proof of Proposition 4

Proposition 4. *Problem 3 is equivalent to finding the robust policy (Eq. 6) for the n-UMDP.*

To prove the proposition we show that the regret Bellman equation for an MDP in Proposition 1 is equivalent to the Bellman equation for the n-MDP. Therefore optimising minimax regret in the original UMDP according to Problem 3 is equivalent to optimising minimax expected cost on the n-UMDP (ie. finding the robust policy for the n-UMDP).

We start by considering the standard robust MDP problem introduced in Equation 6.

$$\pi_{robust} = \operatorname*{argmin}_{\pi\in\Pi}\max_{\sigma^1} V(s_0,\pi), \text{where}$$

$$V(s,\pi) = \sum_{a\in A}\pi(s,a)\cdot[\bar{C}(s,a) + \sum_{s'\in S} T(s,a,s')\cdot V(s',\pi)].$$

Now consider solving the robust MDP problem on the n-UMDP. We need to apply the cost and transition functions from the n-UMDP, and replace actions by options. The adversary now changes the parameters after each option rather than each action.

$$\pi_{robust}(n\text{-}MDP) = \operatorname*{argmin}_{\pi\in\Pi}\max_{\sigma^n} V(s_0,\pi), \text{where} \tag{48}$$

$$V(s,\pi^n) = \sum_{o\in O}\pi^n(s,o)\cdot\left[C^o(s,o) + \sum_{s'\in S} T(s,o,s')\cdot V(s',\pi^n)\right]. \tag{49}$$

To show that this is equivalent to solving Problem 3, we start with the dynamic programming equation for the regret for a policy from Proposition 1

$$reg(s,\pi) = \sum_{a\in A}\pi(s,a)\cdot\left[\bar{C}(s,a) + \sum_{s'\in S} T(s,a,s')\cdot V(s',\pi^*) - V(s,\pi^*) + \sum_{s'\in S} T(s,a,s')\cdot reg(s',\pi)\right].$$

We unroll the definition for n steps, and cancel terms in the same manner as in the proof of Proposition 1.

$$reg(s,\pi) = \sum_{a\in A}\pi(s,a)\cdot\Big[\bar{C}(s,a) - V(s,\pi^*) + \sum_{s'\in S}T(s,a,s')\cdot\Big[\sum_{a\in A}\pi(s',a)\cdot\Big[\bar{C}(s',a)+\dots$$
$$\sum_{s^{n\text{-}1}\in S}T(s^{n\text{-}2},a,s^{n\text{-}1})\cdot\Big[\sum_{a\in A}\pi(s^{n\text{-}1},a)\cdot\Big[\bar{C}(s^{n\text{-}1},a)+\sum_{s^n\in S}T(s^{n\text{-}1},a,s^n)\cdot V(s^n,\pi^*)+\sum_{s^n\in S}T(s^{n\text{-}1},a,s^n)\cdot reg(s^n,\pi)\Big]\Big]\dots\Big]\Big]\Big]. \tag{50}$$

We can substitute policy π for an equivalent option policy, π^n which deterministically chooses an option in each state. At s, π^n chooses a single option o. The associated policy π^o, executes the same action distribution as π over the next n steps.

$$reg(s,\pi^n) = \sum_{o\in O}\pi^n(s,o)\cdot\sum_{a\in A}\pi^o(s,a)\cdot\Big[\bar{C}(s,a) - V(s,\pi^*) + \sum_{s'\in S}T(s,a,s')\cdot\Big[\sum_{a\in A}\pi^o(s',a)\cdot\Big[\bar{C}(s',a)+\dots$$
$$\sum_{s^{n\text{-}1}\in S}T(s^{n\text{-}2},a,s^{n\text{-}1})\cdot\Big[\sum_{a\in A}\pi^o(s^{n\text{-}1},a)\cdot\Big[\bar{C}(s^{n\text{-}1},a)+\sum_{s^n\in S}T(s^{n\text{-}1},a,s^n)\cdot V(s^n,\pi^*)+\sum_{s^n\in S}T(s^{n\text{-}1},a,s^n)\cdot reg(s^n,\pi^n)\Big]\Big]\dots\Big]\Big]\Big]. \tag{51}$$

By Definition 7 of the n-MDP, the expected value of applying π^o for n steps is $V^n(s,\pi^o)$. We can also replace the nested transition functions with $T(s,o,s')$ which by Definition 7 is the transition function for applying an option. Making these substitutions gives

$$reg(s,\pi^n) = \sum_{o\in O}\pi^n(s,o)\cdot\Big[V^n(s,\pi^o) + \sum_{s'\in S}T(s,o,s')\cdot V(s',\pi^*) - V(s,\pi^*) + \sum_{s'\in S}T(s,o,s')\cdot reg(s',\pi^n)\Big]. \tag{52}$$

Substituting the cost function, $C^o(s,o)$ given by Equation 12 we can now write Problem 3 as

$$\pi^n_{reg} = \operatorname*{argmin}_{\pi\in\Pi}\max_{\sigma^n} reg(s_0,\pi^n), \text{where}$$
$$reg(s,\pi^n) = \sum_{o\in O}\pi^n(s,o)\cdot\Big[C^o(s,o) + \sum_{s'\in S}T(s,o,s')\cdot reg(s',\pi^n)\Big].$$

We observe that this statement of Problem 3 is identical to the formulation of the robust policy for the n-UMDP in Equations 48-49. Therefore solving Problem 3 is equivalent to finding the robust policy on the n-UMDP. □

Proof of Proposition 5

Proposition 5. *For dependent uncertainty sets,*

$$\max_{\sigma^n} reg(s_0,\pi) - \max_{\xi_q\in\xi} reg_q(s_0,\pi) \geq 0 \quad \forall\, n\in\mathbb{N},$$
$$\min_{\pi^n}\max_{\sigma^n} reg(s_0,\pi^n) \geq \min_{\pi^{kn}}\max_{\sigma^{kn}} reg(s_0,\pi^{kn}) \quad \forall\, n,k\in\mathbb{N}.$$

For the first part (Equation 15), we observe that the adversary σ^n can apply different combinations of $\xi_q\in\xi$ at each n step interval, and therefore is more powerful than an adversary that chooses only a single $\xi_q\in\xi$. Therefore we have

$$\max_{\sigma^n} reg(s_0,\pi) \geq \max_{\xi_q\in\xi} reg_q(s_0,\pi) \implies \max_{\sigma^n} reg(s_0,\pi) - \max_{\xi_q\in\xi} reg_q(\pi) \geq 0. \tag{53}$$

For the second part (Equation 16), we start with the expression for the regret decomposition for the n-step option MDP from Equation 13.

$$reg(s,\pi^n) = \sum_{o\in O}\pi^n(s,o)\cdot[C^o(s,o) + \sum_{s'\in S}T^o(s,o,s')\cdot reg(s',\pi^n)].$$

Taking the minimax, and then unrolling the regret expression over k sequences of n steps we have

$$\min_{\pi^n}\max_{\sigma^n} reg(s,\pi^n) = \min_{\pi^n}\max_{\xi_1\in\xi}\Big[\sum_{o\in O}\pi^n(s,o)\cdot\Big[C^o_1(s,o) + \sum_{s^n\in S}T^o_1(s,o,s^n)\cdot\Big[\dots$$
$$+\sum_{s^{(k-1)n}\in S}T^o_{k-1}(s^{(k-2)n},o,s^{(k-1)n})\cdot\Big[\max_{\xi_k\in\xi}\Big[\sum_{o\in O}\pi^n(s^{(k-1)n},o)\cdot\Big[C^o_k(s^{(k-1)n},o)+$$
$$\sum_{s^{kn}\in S}T^o_k(s^{(k-1)n},o,s^{kn})\cdot\max_{\sigma^n} reg(s^{kn},\pi^n)\Big]\Big]\Big]\dots\Big]\Big]\Big], \tag{54}$$

where there is a separate maximisation over samples at every n steps as the adversary may change the sample. If instead we were to restrict the adversary to change the parameters at every kn steps the expression would be

$$\min_{\pi^n}\max_{\sigma^{kn}} reg(s,\pi^n) = \min_{\pi^n}\max_{\xi_q\in\xi}\Bigg[\sum_{o\in O}\pi^n(s,o)\cdot\Bigg[C_q^o(s,o)+\sum_{s^n\in S}T_q^o(s,o,s^n)\cdot\Bigg[\dots + \sum_{s^{(k-1)n}\in S}T_q^o(s^{(k-2)n},o,s^{(k-1)n})\cdot\Bigg[\sum_{o\in O}\pi^n(s^{(k-1)n},o)\cdot\Bigg[C_q^o(s^{(k-1)n},o)+ \sum_{s^{kn}\in S}T_q^o(s^{(k-1)n},o,s^{kn})\cdot\max_{\sigma^{kn}} reg(s^{kn},\pi^n)\Bigg]\Bigg]\dots\Bigg]\Bigg]\Bigg]. \quad (55)$$

We observe that for kn step dependence, the adversary performs a single maximisation to choose one sample over the entire sequence of kn steps. This is in contrast to n step dependence, where the adversary is more powerful as it performs a separate maximisation for each of the sequences of n steps. Recursively applying this observation we have that

$$\min_{\pi^n}\max_{\sigma^n} reg(s,\pi^n) \geq \min_{\pi^n}\max_{\sigma^{kn}} reg(s,\pi^n) \quad \forall\, n,k\in\mathbb{N}. \quad (56)$$

Additionally, we have that

$$\min_{\pi^n}\max_{\sigma^{kn}} reg(s,\pi^n) \geq \min_{\pi^{kn}}\max_{\sigma^{kn}} reg(s,\pi^{kn}) \quad \forall\, n,k\in\mathbb{N}, \quad (57)$$

which holds because option policies may be history-dependent. Therefore, a larger number of steps for the option policy, $kn \geq n$, means that each option may consider more of the history, resulting in a more powerful policy.

Combining the inequalities in Equations 56 and 57 we have the required result. □

Constraint Linearisation

Here we describe the process of linearising the nonlinear equality constraints in the optimisation problem in Table 1. We use standard techniques from mathematical programming (see (Williams 2013) for more details). In the case of deterministic policies, the model is solved exactly. For stochastic policies, a linear-piecewise approximation of the original constraints is required.

Deterministic Policies If we assume that π^o is deterministic, then each of the $\pi^o(s,a)$ variables is binary. In this case, we can linearise the constraints in Equations 20 and 23 using a "big M" method. Introducing additional variables denoted c'_q, Equation 20 is replaced by the constraints in Equations 58-61

$$c_q(s,t) = \textstyle\sum_a c'_q(s,a,t) \qquad \forall s\in S^q_{\bar{s},t},\xi_q,t\leq n-1 \quad (58)$$

$$c'_q(s,a,t) \leq c_q(s,a,t) \qquad \forall s\in S^q_{\bar{s},t},a,\xi_q,t\leq n-1 \quad (59)$$

$$c'_q(s,a,t) \leq \pi^o(s,a)\cdot M \qquad \forall s\in S^q_{\bar{s},t},a,\xi_q,t\leq n-1 \quad (60)$$

$$c'_q(s,a,t) \geq c_q(s,a,t) - (1-\pi^o(s,a))\cdot M \qquad \forall s\in S^q_{\bar{s},t},a,\xi_q,t\leq n-1 \quad (61)$$

M is an upper bound on $c_q(s,a,t)$, and is domain-dependent. In a similar manner, Equation 23 is replaced by equivalent constraints on additional variables $V_q^{n'}(s,a,t)$, where M is chosen to be an upper bound on $V_q^n(s,a,t)$.

Stochastic Policies In the case of stochastic policies, the $\pi^o(s,a)$ variables are continuous. The nonlinear constraints in Equation 20 can be approximated by applying separable programming. To convert the model into the appropriate form, we start by introducing additional variables.

$$x_q(s,a,t) = \frac{c_q(s,a,t)+\pi^o(s,a)}{2}$$

$$y_q(s,a,t) = \frac{c_q(s,a,t)-\pi^o(s,a)}{2}$$

Equation 20 can now be written as:

$$c_q(s,t) = \sum_a \left[x_q(s,a,t)^2 - y_q(s,a,t)^2\right] \qquad \forall s\in S^q_{\bar{s},t},\xi_q,t\leq n-1 \quad (62)$$

We apply the common technique from separable programming of approximating the quadratic terms by a piecewise linear function. By backwards induction, we can compute upper and lower bounds on $c_q(s,a,t)$, and therefore on $x_q(s,a,t)$ and $y_q(s,a,t)$. The range for $x_q(s,a,t)$ (and $y_q(s,a,t)$) is divided into m intervals using $m+1$ breakpoints, $\{b_0,b_1,\dots,b_m\}$. We introduce a variable, $\lambda_q^i(s,a,t)$ for each breakpoint, i. We then approximate $x_q(s,a,t)^2$ (and $y_q(s,a,t)^2$) with the following constraints

$$x_q(s,a,t) = \sum_i \lambda_q^i(s,a,t) \cdot b_i \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{63}$$

$$x_q(s,a,t)^2 = \sum_i \lambda_q^i(s,a,t) \cdot b_i^2 \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{64}$$

$$\sum_i \lambda_q^i(s,a,t) = 1 \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{65}$$

$$SOS2(\{\lambda_q^i(s,a,t) | 0 \leq i \leq m\}) \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{66}$$

where $SOS2$ indicates an adjacency constraint whereby at most two variables may have non-zero values, and if two variables are non-zero they must be adjacent. An identical process to that outlined above is applied to approximate and linearise the constraints in Equation 23.

In our experiments, we use 3 breakpoints for the piecewise linear approximation.

Medical Decision Making Domain Details

We adapt the medical decision making domain introduced by Sharma et al. (2019). The state, $(h, d) \in S$ comprises of 2 factors: the health of the patient, $h \in \{0, \ldots, 19\}$, and the day $d \in \{0, \ldots, 6\}$. At each state one of three actions can be applied, each representing different treatments. In each MDP sample the transition probabilities for each treatment differ, corresponding to different responses by patients with different underlying conditions. The health of the patient on the final day determines the cost received.

For each action, the possible relative changes in health are $h' - h = \Delta h \in \{-3, -2, -1, 0, 1, 2, 3\}$. For each health level, a 3×7 matrix representing the likelihood of these 7 possible outcomes conditioned on each of the three actions was created randomly as follows. First, the nominal transition matrix for the UMDP is created by sampling 3 rows of a 7×7 identity matrix. Then, to create each MDP sample we add noise to the nominal transition values. Specifically, the absolute value of Gaussian zero-mean noise with standard deviation 0.1 was added to each value in the matrix, and the rows were then normalised to equal 1.

The cost function is only applied according to the health state on the final day. The cost function was defined by $C(h|d = 6) = 0.05(19 - h) + 2 \cdot \mathbb{1}_{h=0}$, where $\mathbb{1}$ is the indicator function (a large penalty is added for reaching $h = 0$).

Disaster Rescue Domain Details

We adapt this domain from the UMDP literature (Adulyasak et al. 2015; Ahmed et al. 2013, 2017; Bagnell, Ng, and Schneider 2001) to SSP MDPs. An agent navigates an 8-connected grid which contains swamps and obstacles by choosing from 8 corresponding actions. Nominally, for each action the agent transitions to the corresponding target square with $p = 0.8$, and to the two adjacent squares with $p = 0.1$ each. If the target, or adjacent squares are obstacles, the agent transitions to that square with probability 0.05. Any remaining probability mass is assigned to not moving. If the square is a swamp, the cost is sampled uniformly in $[1, 2]$. The cost for entering any other state is 0.5. The agent does not know the exact locations of swamps and obstacles, and instead knows regions where they may be located. To construct a UMDP, each square has a 1/15 chance of being the centre of a swamp region (c_0 and c_1 in Fig. 1), or obstacle region (o_0 in Fig. 1), respectively. The regions include the squares adjacent to the region centres (shaded areas in Fig. 1). To construct a sample, a swamp and obstacle is sampled uniformly from each swamp and obstacle region respectively. Fig. 1 (left) illustrates swamp and obstacle regions for a particular UMDP. Fig. 1 (right) illustrates a possible sample corresponding to the same UMDP.

Underwater Glider Navigation Domain Details

Here we outline the process of creating a UMDP abstraction of underwater glider navigation using ocean current forecasts, based on the approach from (Liu and Sukhatme 2018). For each UMDP, we randomly sample a region of the Norwegian sea within the boundaries of 61.1° to 61.2° latitude and 4.5° to 4.65° longitude. The region is discretised into grid cells with a side length of L. Each grid cell is associated with a state in the MDP abstraction, s. We write $\mathbf{x}_s$ to denote the position of the centre of the grid cell associated with s. We can map any position to discrete space: $\mathbf{x} \to s$ if $||\mathbf{x} - \mathbf{x}_s||_\infty < L/2$. Each action corresponds to a heading direction for the glider.

For each MDP sample, we repeat the following process to generate an MDP corresponding the time of the day in hourly intervals between 6am and 6pm. We denote the velocity of the glider relative to the water when taking action a by $\mathbf{v}_g(a)$. The velocity of the ocean current at state s is denoted $\mathbf{v}_c(s)$. This value is found by referring to the ocean current forecast for the appropriate time of day, which is available online.* The expected position of the glider after applying action a in state s is

$$\mathbb{E}[\mathbf{x}'|s,a] = \mathbf{x}_s + (\mathbf{v}_g(a) + \mathbf{v}_c(s)) \cdot \Delta t,$$

where Δt is the time between each time step in the MDP abstraction. The resulting position of the glider is also subject to noise, $\mathbf{d}(s,a)$. This noise is due to multiple sources such as: heading tracking error of the glider, environmental disturbances, and the

*https://marine.copernicus.eu/

glider not starting the action exactly at $\mathbf{x}_s$. We model $\mathbf{d}(s, a)$ with Gaussian noise with covariance matrix $\mathbf{\Sigma}$ for all states and actions. Thus,

$$\mathbf{x}'|_{s,a} \sim \mathcal{N}(\mathbb{E}[\mathbf{x}'|s, a], \mathbf{\Sigma})$$

The transition probabilities in the MDP abstraction can be found by integrating this distribution over each of the grid cells. In our experiments we use forecast data for May 1st 2020 and the following values for the abstraction:

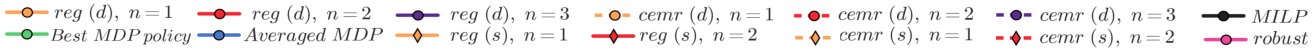
- L = 500m
- $||\mathbf{v}_g(a)||_2 =$ 0.6m/s for all actions
- $\Delta t = 800s$
- $\mathbf{\Sigma} = \text{diag}(150^2, 150^2)$
- There are 12 heading directions (actions) evenly spaced over 360°

p-Values for Experimental Results

To assess the statistical significance of the difference in performance between each of the methods, we include a table of p-values for each of the methods. The top row of each table shows the mean and standard deviation of the normalised maximum regret for each of the methods across all of the runs. The remainder of the table includes p-values, which can be interpreted as follows. The p-value in the row of Method 1, and the column of Method 2 is the p-value for the hypothesis that Method 1 has better average performance than Method 2, calculated using a standard two-sample t-test. If the results for Method 1 do not have a better (lower) average than Method 2, then no p-value is included.

To compute p-values for the disaster rescue and glider domain where we tested a number of problem sizes, we first average the normalised maximum regret across all runs of all problem sizes, and compute the associated standard deviation. This mean and standard deviation is used to compute the p-values. We only include methods which solved all problem sizes within the 600s time limit. Similarly, for the medical domain we include methods which were within the 600s time limit.

Disaster rescue domain



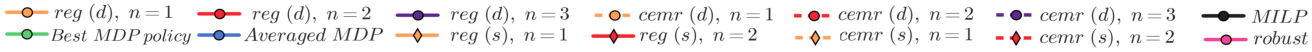
Method											
max reg	0.576; 0.25	0.340; 0.19	**0.297**; 0.17	0.840; 0.25	0.633; 0.25	0.521; 0.24	0.647; 0.25	0.661; 0.28	0.578; 0.27	0.353; 0.18	0.695; 0.23
	-	-	-	< 0.0001	0.017	-	0.004	0.001	0.47	-	< 0.0001
	< 0.0001	-	-	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	0.26	< 0.0001
	< 0.0001	0.014	-	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	0.0015	< 0.0001
	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	< 0.0001	-	-	0.30	0.16	-	-	0.008
	0.018	-	-	< 0.0001	< 0.0001	-	< 0.0001	< 0.0001	0.019	-	< 0.0001
	-	-	-	< 0.0001	-	-	-	0.31	-	-	0.031
	-	-	-	< 0.0001	-	-	-	-	-	-	0.11
	-	-	-	< 0.0001	0.049	-	0.007	0.003	-	-	< 0.0001
	< 0.0001	-	-	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-	< 0.0001
	-	-	-	< 0.0001	-	-	-	-	-	-	-

Table 3: Top row contains mean normalised maximum regret for disaster rescue domain across all problem sizes, in the format: mean; standard deviation. The remainder of the table contains p-values for the comparisons between each method. Methods which did not find a solution for all problem sizes within the 600s time limit are not included.

Method											
max reg	0.601; 0.28	0.410; 0.22	**0.348**; 0.20	0.826; 0.25	0.636; 0.26	0.528; 0.24	0.680; 0.28	0.643; 0.28	0.626; 0.31	0.389; 0.21	0.669; 0.26
	-	-	-	< 0.0001	-	-	0.004	0.081	0.22	-	0.010
	<0.0001	-	-	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	-	<0.0001
	<0.0001	0.003	-	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	0.031	<0.0001
	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	<0.0001	-	-	0.064	0.40	-	-	0.12
	0.005	-	-	<0.0001	<0.0001	-	<0.0001	<0.0001	<0.0001	-	<0.0001
	-	-	-	<0.0001	-	-	-	-	-	-	-
	-	-	-	<0.0001	-	-	0.11	-	-	-	0.18
	-	-	-	<0.0001	0.37	-	0.044	0.30	-	-	0.080
	<0.0001	0.18	-	-	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	-	<0.0001
	-	-	-	<0.0001	-	-	0.35	-	-	-	-

Table 4: Top row contains mean normalised maximum regret for disaster rescue domain on the test set across all problem sizes, in the format: mean; standard deviation. The remainder of the table contains p-values for the comparisons between each method. Methods which did not find a solution for all problem sizes within the 600s time limit are not included.

Underwater glider domain



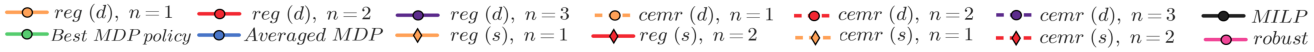
Method											
max reg	0.590; 0.25	0.537; 0.22	**0.504**; 0.21	0.813; 0.21	0.767; 0.20	0.696; 0.19	0.566; 0.23	0.652; 0.25	0.681; 0.27	0.557; 0.21	0.802; 0.20
	-	-	-	<0.0001	<0.0001	<0.0001	-	0.016	0.001	-	<0.0001
	0.026	-	-	<0.0001	<0.0001	<0.0001	0.13	<0.0001	<0.0001	0.21	<0.0001
	0.0007	0.092	-	<0.0001	<0.0001	<0.0001	0.0075	<0.0001	<0.0001	0.015	<0.0001
	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	0.026	-	-	-	-	-	-	0.065
	-	-	-	<0.0001	-	-	-	-	-	-	<0.0001
	0.19	-	-	<0.0001	<0.0001	<0.0001	-	0.001	<0.0001	-	-
	-	-	-	<0.0001	<0.0001	0.044	-	-	0.17	-	-
	-	-	-	<0.0001	0.001	0.29	-	-	-	-	-
	0.11	-	-	-	-	<0.0001	0.36	0.0002	-	-	-
	-	-	-	0.32	-	-	-	-	-	-	-

Table 5: Top row contains mean normalised maximum regret for underwater glider domain across all problem sizes, in the format: mean; standard deviation. The remainder of the table contains p-values for the comparisons between each method. Methods which did not find a solution for all problem sizes within the 600s time limit are not included.

Method											
max reg	0.715; 0.24	0.679; 0.23	**0.652**; 0.23	0.849; 0.21	0.812; 0.20	0.757; 0.19	0.688; 0.24	0.718; 0.23	0.758; 0.24	0.680; 0.21	0.827; 0.19
	-	-	-	<0.0001	0.0001	0.047	-	0.46	0.061	-	<0.0001
	0.093	-	-	<0.0001	<0.0001	0.0008	0.37	0.072	0.002	0.48	<0.0001
	0.011	0.16	-	<0.0001	<0.0001	<0.0001	0.093	0.007	<0.0001	0.14	<0.0001
	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	0.060	-	-	-	-	-	-	0.25
	-	-	-	0.0001	0.008	-	-	-	0.48	-	0.0008
	0.17	-	-	<0.0001	<0.0001	0.004	-	0.13	0.006	-	-
	-	-	-	<0.0001	0.0001	0.055	-	-	0.071	-	<0.0001
	-	-	-	0.0003	0.018	-	-	-	-	-	0.003
	0.090	-	-	<0.0001	<0.0001	0.0005	0.38	0.068	0.002	-	<0.0001
	-	-	-	0.17	-	-	-	-	-	-	-

Table 6: Top row contains mean normalised maximum regret for the underwater glider domain on the test set across all problem sizes, in the format: mean; standard deviation. The remainder of the table contains p-values for the comparisons between each method. Methods which did not find a solution for all problem sizes within the 600s time limit are not included.

Medical decision making domain



Method											
max reg	0.596; 0.22	0.538; 0.18	**0.497**; 0.16	0.906; 0.11	0.885; 0.13	0.880; 0.12	0.557; 0.20	0.596; 0.22	0.641; 0.23	0.529; 0.15	0.846; 0.12
	-	-	-	<0.0001	<0.0001	<0.0001	-	-	0.013	-	<0.0001
	0.0007	-	-	-	-	-	0.13	0.0007	<0.0001	-	<0.0001
	-	0.004	-	-	-	-	0.0001	<0.0001	<0.0001	-	<0.0001
	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	0.025	-	-	-	-	-	-	-
	-	-	-	0.006	0.33	-	-	-	-	-	-
	0.019	-	-	-	-	-	-	0.019	<0.0001	-	<0.0001
	-	-	-	<0.0001	<0.0001	<0.0001	-	-	0.013	-	<0.0001
	-	-	-	<0.0001	<0.0001	<0.0001	-	-	-	-	<0.0001
	<0.0001	0.27	-	<0.0001	<0.0001	<0.0001	0.039	<0.0001	<0.0001	-	<0.0001
	-	-	-	<0.0001	0.0003	0.0008	-	-	-	-	-

Table 7: Top row contains mean normalised maximum regret for medical decision making domain over all 250 runs, in the format: mean; standard deviation. The remainder of the table contains p-values for the comparisons between each method. Methods which did not find a solution for all problem sizes within the 600s time limit are not included.

Method											
max reg	0.674; 0.19	0.636; 0.18	0.625; 0.18	0.870; 0.13	0.855; 0.14	0.852; 0.13	0.706; 0.20	0.677; 0.20	0.715; 0.19	**0.574**; 0.12	0.814;0.13
	-	-	-	<0.0001	<0.0001	<0.0001	0.034	0.43	0.008	-	<0.0001
	0.011	-	-	<0.0001	<0.0001	<0.0001	<0.0001	0.008	<0.0001	-	<0.0001
	0.0016	0.25	-	<0.0001	<0.0001	<0.0001	<0.0001	0.0012	<0.0001	-	-
	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	0.11	-	-	-	-	-	-	-
	-	-	-	0.061	0.40	-	-	-	-	-	-
	-	-	-	<0.0001	<0.0001	<0.0001	-	-	0.30	-	<0.0001
	-	-	-	<0.0001	<0.0001	<0.0001	0.053	-	0.015	-	<0.0001
	-	-	-	<0.0001	<0.0001	<0.0001	-	-	-	-	<0.0001
		<0.0001	<0.0001	0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	-	<0.0001
	-	-	-	<0.0001	0.0004	0.0009	-	-	-	-	-

Table 8: Top row contains mean normalised maximum regret on the test set for medical decision making domain over all 250 runs, in the format: mean; standard deviation. The remainder of the table contains p-values for the comparisons between each method. Methods which did not find a solution for all problem sizes within the 600s time limit are not included.

Appendices of Chapter 5

A Proof of Proposition 1

Let $\mathcal{M}^+$ be a BAMDP. Then:

$$\text{CVaR}_\alpha(G_\pi^{\mathcal{M}^+}) = \min_{\substack{\delta:\mathcal{T}\to\mathbb{R}_{\geq 0}\\ \xi\in\Xi}} \mathbb{E}\big[G_\pi^{\mathcal{M}^+_{\delta,\xi}}\big], \quad \textbf{s.t.} \tag{14}$$

$$0 \leq \delta(T_p)\xi_p(s_0,a_0,s_1)\xi_p(s_1,a_1,s_2)\cdots\xi_p(s_{H\text{-}1},a_{H\text{-}1},s_H) \leq \frac{1}{\alpha},$$

$$\forall\, T_p \in \mathcal{T},\ \forall (s_0,a_0,s_1,a_1,\ldots,s_H) \in \mathcal{H}_H.$$

Proof: We denote by $\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)$ the probability of history h_H in BAMDP $\mathcal{M}^+$, under policy π. This probability can be expressed as

$$\mathcal{P}_\pi^{\mathcal{M}^+}(h_H) = T^+\big((s_0,h_0),\pi(h_0),(s_1,h_1)\big)T^+\big((s_1,h_1),\pi(h_1),(s_2,h_2)\big)\ldots T^+\big((s_{H\text{-}1},h_{H\text{-}1}),\pi(h_{H\text{-}1}),(s_H,h_H)\big). \tag{15}$$

By substituting in the transition function for the BAMDP from Equation 7 we have

$$\mathcal{P}_\pi^{\mathcal{M}^+}(h_H) = \int_T T(s_0,\pi(h_0),s_1)\mathcal{P}(T|h_0)dT \cdot \int_T T(s_1,\pi(h_1),s_2)\mathcal{P}(T|h_1)dT\ldots \int_T T(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H)\mathcal{P}(T|h_{H-1})dT. \tag{16}$$

Bayes' rule states that

$$\mathcal{P}(T|h_1) = \frac{T(s_0,\pi(h_0),s_1)\cdot\mathcal{P}(T|h_0)}{\int_{T'} T'(s_0,\pi(h_0),s_1)\cdot\mathcal{P}(T'|h_0)dT'}. \tag{17}$$

Substituting Bayes' posterior update into Equation 16 we have

$$\mathcal{P}_\pi^{\mathcal{M}^+}(h_H) = \int_T T(s_0,\pi(h_0),s_1)\mathcal{P}(T|h_0)dT\cdot\int_T T(s_1,\pi(h_1),s_2)\frac{T(s_0,\pi(h_0),s_1)\cdot\mathcal{P}(T|h_0)}{\int_{T'} T'(s_0,\pi(h_0),s_1)\cdot\mathcal{P}(T'|h_0)dT'}dT\ldots$$
$$\int_T T(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H)\mathcal{P}(T|h_{H-1})dT = \int_T T(s_0,\pi(h_0),s_1)\cdot T(s_1,\pi(h_1),s_2)\cdot\mathcal{P}(T|h_0)dT\ldots \int_T T(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H)\mathcal{P}(T|h_{H-1})dT. \tag{18}$$

Repeating this process of substitution and cancellation we get

$$\mathcal{P}_\pi^{\mathcal{M}^+}(h_H) = \left[\int_T T(s_0,\pi(h_0),s_1)\cdot T(s_1,\pi(h_1),s_2)\ldots T(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H)\cdot\mathcal{P}(T|h_0)dT\right]. \tag{19}$$

Define $\mathcal{T}$ to be the support of $\mathcal{P}(T|h_0)$, and let $T_p \in \mathcal{T}$ be a plausible transition function. Now consider a perturbation of the prior distribution over transition functions $\delta : \mathcal{T} \to \mathbb{R}_{\geq 0}$ such that $\int_{T_p}\delta(T_p)\mathcal{P}(T_p|h_0)dT_p = 1$. Additionally, for $T_p \in \mathcal{T}$, consider a perturbation of T_p, $\xi_p : S\times A\times S \to \mathbb{R}_{\geq 0}$ such that $\sum_{s'\in S}\xi_p(s,a,s')\cdot T_p(s,a,s') = 1\ \ \forall s,a$. We denote the set of all possible perturbations of T_p as Ξ_p, and the set of all perturbations over $\mathcal{T}$ as $\Xi = \times_{T_p\in\mathcal{T}}\Xi_p$. For BAMDP $\mathcal{M}^+$, $\delta : \mathcal{T}\to\mathbb{R}_{\geq 0}$, and $\xi\in\Xi$, we define $\mathcal{M}^+_{\delta,\xi}$ as the BAMDP, obtained from $\mathcal{M}^+$, with perturbed prior distribution $\mathcal{P}^\delta(T_p|h_0) = \delta(T_p)\cdot\mathcal{P}(T_p|h_0)$ and perturbed transition functions

$T_p^{\xi}(s,a,s') = \xi_p(s,a,s') \cdot T_p(s,a,s')$. We denote by $\mathcal{P}_\pi^{\mathcal{M}^+_{\delta,\xi}}(h_H)$ the probability of h_H in the perturbed BAMDP, $\mathcal{M}^+_{\delta,\xi}$, which can be expressed as follows

$$\begin{aligned}
\mathcal{P}_\pi^{\mathcal{M}^+_{\delta,\xi}}(h_H) &= \left[\int_{T_p} T_p^{\xi}(s_0,\pi(h_0),s_1) \cdot T_p^{\xi}(s_1,\pi(h_1),s_2)) \dots T_p^{\xi}(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H) \cdot \mathcal{P}^{\delta}(T_p|h_0) dT_p \right] \\
&= \left[\int_{T_p} \xi_p(s_0,\pi(h_0),s_1) \cdot T_p(s_0,\pi(h_0),s_1) \cdot \xi_p(s_1,\pi(h_1),s_2) \cdot T_p(s_1,\pi(h_1),s_2)) \dots \right. \\
&\qquad \left. \cdot \xi_p(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H) \cdot T_p(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H) \cdot \delta(T_p) \cdot \mathcal{P}(T_p|h_0) dT_p \right] \\
&= \left[\int_{T_p} \xi_p(s_0,\pi(h_0),s_1) \cdot \xi_p(s_1,\pi(h_1),s_2) \dots \xi_p(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H) \cdot \delta(T_p) \cdot \right. \\
&\qquad \left. T_p(s_0,\pi(h_0),s_1) \cdot T_p(s_1,\pi(h_1),s_2)) \dots T_p(s_{H\text{-}1},\pi(h_{H\text{-}1}),s_H) \cdot \mathcal{P}(T_p|h_0) dT_p \right] \quad (20)
\end{aligned}$$

Let $\kappa_{\delta,\xi}(h_H) = \frac{\mathcal{P}_\pi^{\mathcal{M}^+_{\delta,\xi}}(h_H)}{\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)}$ denote the total perturbation to the probability of any path. Provided that the perturbations satisfy the condition

$$\begin{gathered}
0 \le \delta(T_p)\xi_p(s_0,a_0,s_1)\xi_p(s_1,a_1,s_2) \cdots \xi_p(s_{H\text{-}1},a_{H\text{-}1},s_H) \le \frac{1}{\alpha}, \\
\forall\, T_p \in \mathcal{T},\ \forall (s_0,a_0,s_1,a_1,\dots,s_H) \in \mathcal{H}_H,
\end{gathered}$$

we observe from Equation 20 that the perturbation to the probability of any path is bounded by

$$0 \le \kappa_{\delta,\xi}(h_H) \le \frac{1}{\alpha}. \quad (21)$$

The expected value of the perturbation to the probability of any path is

$$\begin{aligned}
\mathbb{E}[\kappa_{\delta,\xi}(h_H)] = \sum_{h_H \in \mathcal{H}_H} \left[\mathcal{P}_\pi^{\mathcal{M}^+}(h_H) \cdot \kappa_{\delta,\xi}(h_H) \right] &= \sum_{h_H \in \mathcal{H}_H} \left[\mathcal{P}_\pi^{\mathcal{M}^+}(h_H) \frac{\mathcal{P}_\pi^{\mathcal{M}^+_{\delta,\xi}}(h_H)}{\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)} \right] \\
&= \sum_{h_H \in \mathcal{H}_H} \left[\mathcal{P}_\pi^{\mathcal{M}^+_{\delta,\xi}}(h_H) \right] = 1, \quad (22)
\end{aligned}$$

where the last equality holds because the conditions $\sum_{s' \in S} \xi_p(s,a,s') \cdot T_p(s,a,s') = 1 \ \ \forall s,a,p$ and $\int_{T_p} \delta(T_p)\mathcal{P}(T_p|h_0)dT_p = 1$ ensure that the distribution over paths in the perturbed BAMDP, $\mathcal{M}^+_{\delta,\xi}$, is a valid probability distribution. Therefore,

$$\begin{aligned}
\min_{\substack{\delta:\mathcal{T}\to\mathbb{R}_{\ge 0} \\ \xi \in \Xi}} \mathbb{E}\left[G_\pi^{\mathcal{M}^+_{\delta,\xi}}\right] = \min_{\substack{\delta:\mathcal{T}\to\mathbb{R}_{\ge 0} \\ \xi \in \Xi}} \sum_{h_H} \left[\mathcal{P}_\pi^{\mathcal{M}^+_{\delta,\xi}}(h_H) \cdot \widetilde{r}(h_H) \right] = \\
\min_{\substack{\kappa_{\delta,\xi},\ \textbf{s.t.} \\ 0 \le \kappa_{\delta,\xi}(h_H) \le \frac{1}{\alpha} \\ \mathbb{E}[\kappa_{\delta,\xi}(h_H)] = 1}} \sum_{h_H} \left[\mathcal{P}_\pi^{\mathcal{M}^+}(h_H) \cdot \kappa_{\delta,\xi}(h_H) \cdot \widetilde{r}(h_H) \right] = \text{CVaR}_\alpha(G_\pi^{\mathcal{M}^+}) \quad (23)
\end{aligned}$$

where $\widetilde{r}(h_H)$ indicates the total sum of rewards for history h_H, and the last equality holds from the CVaR dual representation theorem stated in Equations 2 and 3 [34, 37]. □

B Proof of Proposition 2

Formulation of CVaR optimisation in BAMDPs as a stochastic game.

$$\max_{\pi\in\Pi^{\mathcal{M}^+}} \text{CVaR}_\alpha(G_\pi^{\mathcal{M}^+}) = \max_{\pi\in\Pi^{\mathcal{G}^+}} \min_{\sigma\in\Sigma^{\mathcal{G}^+}} \mathbb{E}\big[G_{(\pi,\sigma)}^{\mathcal{G}^+}\big]. \tag{24}$$

Proof: Proposition 2 directly extends Proposition 1 in [8] to BAMDPs. Let $\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)$ denote the probability of history h_H in BAMDP $\mathcal{M}^+$ under policy π as defined in Equation 15. We denote by $\mathcal{P}_{\pi,\sigma}^{\mathcal{G}^+}(h_H)$ the probability of history h_H by following policy π and adversary perturbation policy σ in BA-CVaR-SG, $\mathcal{G}^+$. This probability can be expressed as

$$\begin{aligned}\mathcal{P}_{\pi,\sigma}^{\mathcal{G}^+}(h_H) = T^+\big((s_0,h_0),\pi(h_0),(s_1,h_1)\big)\cdot\xi\big((s_0,h_0),\pi(h_0),s_1\big)\cdot T^+\big((s_1,h_1),\pi(h_1),(s_2,h_2)\big)\cdot\xi\big((s_1,h_1),\pi(h_1),s_2\big)\\ \dots T^+\big((s_{H\text{-}1},h_{H\text{-}1}),\pi(h_{H\text{-}1}),(s_H,h_H)\big)\cdot\xi\big((s_{H\text{-}1},h_{H\text{-}1}),\pi(h_{H\text{-}1}),s_H\big),\end{aligned} \tag{25}$$

where $\xi\big((s,h),a,s'\big)$ indicates the perturbation applied by σ for successor state s' after action a is executed in augmented state (s,h).

Let $\kappa_\sigma(h_H)$ denote the total perturbation by the adversary to the probability of any history in the BAMDP,

$$\kappa_\sigma(h_H) = \frac{\mathcal{P}_{\pi,\sigma}^{\mathcal{G}^+}(h_H)}{\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)} = \xi\big((s_0,h_0),\pi(h_0),s_1\big)\cdot\xi\big((s_1,h_1),\pi(h_1),s_2\big)\dots\xi\big((s_{H\text{-}1},h_{H\text{-}1}),\pi(h_{H\text{-}1}),s_H\big). \tag{26}$$

By the definition of the admissible adversary perturbations in Equation 11 the perturbation to the probability of any path is bounded by

$$0 \le \kappa_\sigma(h_H) \le \frac{1}{\alpha}. \tag{27}$$

The expected value of the perturbation to the probability of any path is

$$\begin{aligned}\mathbb{E}[\kappa_\sigma(h_H)] = \sum_{h_H\in\mathcal{H}_H}\Big[\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)\cdot\kappa_\sigma(h_H)\Big] = \sum_{h_H\in\mathcal{H}_H}\Big[\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)\frac{\mathcal{P}_{\pi,\sigma}^{\mathcal{G}^+}(h_H)}{\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)}\Big]\\ = \sum_{h_H\in\mathcal{H}_H}\Big[\mathcal{P}_{\pi,\sigma}^{\mathcal{G}^+}(h_H)\Big] = 1,\end{aligned} \tag{28}$$

where the last equality holds because Equation 11 ensures that the perturbed transition probabilities are a valid probability distribution. Therefore, the perturbed distribution over histories is also a valid probability distribution. Therefore,

$$\begin{aligned}\max_{\pi\in\Pi^{\mathcal{G}^+}}\min_{\sigma\in\Sigma^{\mathcal{G}^+}}\mathbb{E}\big[G_{(\pi,\sigma)}^{\mathcal{G}^+}\big] = \max_{\pi\in\Pi^{\mathcal{G}^+}}\min_{\sigma\in\Sigma^{\mathcal{G}^+}}\sum_{h_H\in\mathcal{H}_H}\Big[\mathcal{P}_{\pi,\sigma}^{\mathcal{G}^+}(h_H)\cdot\widetilde{r}(h_H)\Big] =\\ \max_{\pi\in\Pi^{\mathcal{G}^+}}\min_{\substack{\kappa_\sigma,\ \textbf{s.t.}\\ 0\le\kappa_\sigma(h_H)\le\frac{1}{\alpha}\\ \mathbb{E}[\kappa_\sigma(h_H)]=1}}\sum_{h_H}\Big[\mathcal{P}_\pi^{\mathcal{M}^+}(h_H)\cdot\kappa_\sigma(h_H)\cdot\widetilde{r}(h_H)\Big] = \max_{\pi\in\Pi^{\mathcal{G}^+}}\text{CVaR}_\alpha(G_\pi^{\mathcal{M}^+}).\end{aligned} \tag{29}$$

where $\widetilde{r}(h_H)$ indicates the total sum of rewards for history h_H, and the last equality holds from the CVaR dual representation theorem stated in Equations 2 and 3 [34, 37]. □

C Proof of Proposition 3

Let $\delta \in (0,1)$. Provided that c_{bo} is chosen appropriately (details in the appendix), as the number of perturbations expanded approaches ∞, a perturbation within any $\epsilon > 0$ of the optimal perturbation will be expanded by the Bayesian optimisation procedure with probability $\geq 1-\delta$.

Proof: Consider an adversary decision node, v, associated with augmented state (s, ha, y) in the BA-CVaR-SG. Let $Q((s, ha, y), \xi)$ denote the true minimax optimal value of applying adversary perturbation ξ at augmented state (s, ha, y) in the BA-CVaR-SG. We begin by proving that $Q((s, ha, y), \xi)$ is continuous with respect to ξ. Define a function $d : S \to \mathbb{R}$, such that $\xi + d$ produces a valid adversary perturbation. We can write $Q((s, ha, y), \xi + d)$ as a sum over successor states

$$Q((s, ha, y), \xi + d) = \sum_{s'} (\xi(s') + d(s')) \cdot T^{+}((s, h), a, (s', has')) \cdot V(s', has', y \cdot (\xi(s') + d(s'))), \tag{30}$$

where $V(s, h, y)$ denotes the optimal expected value in the BA-CVaR-SG at augmented state (s, h, y). Because $V(s, h, y)$ equals the CVaR at confidence level y when starting from (s, h), and CVaR is continuous with respect to the confidence level [34], we have that

$$\begin{aligned} \lim_{(d(s_1),\ldots,d(s_{|S|}))\to(0,\ldots,0)} Q((s, ha, y), \xi + d) &= \sum_{s'} \xi(s') \cdot T^{+}((s, h), a, (s', has')) \cdot V(s', has', y \cdot \xi(s')) \\ &= Q((s, ha, y), \xi). \end{aligned} \tag{31}$$

Therefore, $Q((s, ha, y), \xi)$ is continuous with respect to ξ. Now, consider the Gaussian kernel which we use for Bayesian optimisation, $k : \Xi \times \Xi \to \mathbb{R}$, where Ξ is the compact set of admissible adversary actions according to Equation 11

$$k(\xi, \xi') = \exp\Big(- \frac{||\xi - \xi'||^2}{2l^2} \Big). \tag{32}$$

Because the Gaussian kernel is a "universal" kernel [23], the reproducing kernel Hilbert space (RKHS), $\mathcal{H}_k$, corresponding to kernel k is dense in $\mathcal{C}(\Xi)$, the set of real-valued continuous functions on Ξ. Because $Q((s, ha, y), \xi)$ is continuous with ξ, $Q((s, ha, y), \xi)$ belongs to the RKHS of k. Because the true underlying function that we are optimising belongs to the RKHS of the kernel, Theorem 3 from [41] applies.

Consider minimising the true underlying function $Q((s, ha, y), \xi)$ by using GP Bayesian optimisation to decide the next adversary perturbation $\xi \in \Xi$ to sample. Our prior is $GP(0, k(\xi, \xi'))$, and noise model $N(0, \sigma_n^2)$. For each perturbation expanded so far at v, $\xi \in \widetilde{\Xi}(v)$, we assume that the Q value of ξ estimated using MCTS, $\widehat{Q}((s, ha, y), \xi)$, is a noisy estimate of the true optimal Q value, i.e. $\widehat{Q}((s, ha, y), \xi) = Q((s, ha, y), \xi) + \epsilon_n$, where the noise is bounded by $\epsilon_n < \sigma_n$. For each round, $t = 1, 2, \ldots$, we choose the new perturbation to be expanded using a lower confidence bound, $\xi_t = \arg\min_{\xi}[\mu_{t-1}(\xi) - c_{bo}\sigma_{t-1}(\xi)]$, where $\mu_{t-1}(\xi)$, and $\sigma_{t-1}(\xi)$ are the mean and standard deviation of the GP posterior distribution after performing Bayesian updates using the Q-value estimates, $\widehat{Q}((s, ha, y), \xi)$, up to $t-1$.

We define the cumulative regret after T rounds of expanding new perturbation actions as

$$R_T = \sum_{t=1}^{T} Q((s, ha, y), \xi_t) - Q((s, ha, y), \xi^*), \tag{33}$$

where ξ^* is the optimal adversary perturbation at (s, ha, y). Theorem 3 from [41] states the following. Let $\delta \in (0,1)$. Assume $||Q((s, ha, y), \cdot)||_k^2 \leq B$, where $||f||_k$ denotes the RKHS norm of f induced by k. Set the exploration parameter in the lower confidence bound acquisition function

to $c_{bo} = \sqrt{2B + 300\gamma_t \log(t/\delta)^3}$, where γ_t is the maximum information gain at round t, which is defined and bounded in [41]. Then the following holds

$$\Pr\left\{R_T \leq \sqrt{C_1 T \beta_T \gamma_T} \quad \forall T \geq 1\right\} \geq 1 - \delta, \tag{34}$$

where $C_1 = 8/\log(1 + \sigma_n^2)$. Observe that Equation 34 implies that with probability of at least $1 - \delta$ the cumulative regret is sublinear, i.e. $\lim_{T\to\infty} R_T/T = 0$. Now we complete the proof via contradiction. Consider any $\varepsilon > 0$. Suppose that with probability greater than δ, $\lim_{T\to\infty} Q((s, ha, y), \xi_T) \geq Q((s, ha, y), \xi^*) + \epsilon$. If this is the case, then with probability greater than δ the regret will be linear with T, contradicting Theorem 3 from [41]. This completes the proof that for any $\epsilon > 0$, $\lim_{T\to\infty} Q((s, ha, y), \xi_T) < Q((s, ha, y), \xi^*) + \epsilon$ with probability of at least $1 - \delta$. □

In our experiments, we used $c_{bo} = 2$ as described in Section 5. The GP parameters that used in the experiments are described in Section D.2.

D Additional Experiment Details

D.1 Autonomous Car Navigation Domain

An autonomous car must navigate along roads to a destination as illustrated in Figure 1 by choosing from actions *{up, down, left, right}*. There are four types of roads: *{highway, main road, street, lane}* with different transition duration characteristics. The transition duration distribution for each type of road is unknown. Instead, the prior belief over the transition duration distribution for each road type is modelled by a Dirichlet distribution. Thus, the belief is represented by four Dirichlet distributions. Upon traversing each section of road, the agent receives reward $-time$ where $time$ is the transition duration for traversing that section of road. Upon reaching the destination, the agent receives a reward of 80.

For each type of road, it is assumed that there are three possible outcomes for the transition to the next junction, $\{fast,\ medium,\ slow\}$. For each type of road, the prior parameters for the Dirichlet distribution are $\{\alpha_{fast} = 1,\ \alpha_{medium} = 1,\ \alpha_{slow} = 0.4\}$. The transition duration associated with each outcome varies between each of the road types as follows:

- *highway*: $\{time_{fast} = 1,\ time_{medium} = 2,\ time_{slow} = 18\}$
- *main road*: $\{time_{fast} = 2,\ time_{medium} = 4,\ time_{slow} = 13\}$
- *street*: $\{time_{fast} = 4,\ time_{medium} = 5,\ time_{slow} = 11\}$
- *lane*: $\{time_{fast} = 7,\ time_{medium} = 7,\ time_{slow} = 8\}$

The prior over transition duration distributions implies a tradeoff between risk and expected reward. According to the prior, the *highway* road type is the fastest in expectation, but there is a risk of incurring very long durations. The *lane* road type is the slowest in expectation, but there is no possibility of very long durations. The *main road* and *street* types are in between these two extremes.

After executing a transition along a road and receiving reward $-time$, the agent updates the Dirichlet distribution associated with that road type, refining its belief over the transition duration distribution for that road type.

D.2 Gaussian Process Details

We found that optimising the GP hyperparameters online at each node was overly computationally demanding. Instead, for all experiments we set the observation noise to $\sigma_n^2 = 1$ and the length scale to $l = \frac{1}{5y}$, where y is the state factor representing the adversary budget. These parameters were found to work well empirically.

The prior mean for the GP was set to zero, which is optimistic for the minimising adversary. The Shogun Machine Learning toolbox [40] was used for GP regression.

D.3 Policy Gradient Method Details

The method *CVaR BAMDP PG* applies the CVaR gradient policy update from [45] with the vectorised belief representation for BAMDPs with linear function approximation proposed in ([15], Section 3.3). The function approximation enables generalisation between similar beliefs.

Following the belief representation proposed by [15], we draw a set M of MDP samples from the root belief. For each sample, $T_m \in M$, we associate a particle weight $z_m(h)$. The vector $z(h)$ containing the particle weights is a finite-dimensional approximate representation of the belief. The weights are initialised to be $z_m(h_0) = \frac{1}{|M|}$. During each simulation, as transitions are observed the particle weights are updated, $z_m(has') \propto T_m(s, a, s')z_m(h)$. We combine the belief feature vector, $z(h)$, with a feature vector representing the state-action pair, $\phi(s, a)$, in a bilinear form

$$F(h, s, a, \mathbf{W}) = z(h)^T \mathbf{W} \phi(s, a) \tag{35}$$

where $\mathbf{W}$ is a learnable parameter matrix. As we address problems with finite state and action spaces, for the state-action feature vector, $\phi(s, a)$, we chose to use a binary feature for each state-action pair. The action weighting, $F(h, s, a, \mathbf{W})$ defines a softmax stochastic action selection policy

$$f(a|h, s, \mathbf{W}) = \frac{\exp(F(h, s, a, \mathbf{W}))}{\sum_{a'} \exp(F(h, s, a', \mathbf{W}))} \tag{36}$$

where $f(a|h, s, \mathbf{W})$ is the probability of choosing action a at history h, state s, and current parameter matrix $\mathbf{W}$. To perform a simulation, we sample a model from the prior belief using root sampling [15] and simulate an episode by choosing actions according to $f(a|h, s, \mathbf{W})$.

After performing a minibatch of simulations, we update the parameter matrix

$$\mathbf{W} \leftarrow \mathbf{W} + \lambda \nabla_{\text{CVaR}} \tag{37}$$

where ∇_{CVaR} is the Monte Carlo estimate of the CVaR gradient from ([45], Equation 6) and λ is the learning rate.

We use $|M| = 25$, as this number of particles performed well in [15]. Following [45], we used a minibatch size of 1000 simulations for each update. To initialise the parameter matrix, we first computed the policy for the *CVaR VI Expected MDP* method. We denote by $\mathbf{W}(s, a, m)$ the parameter matrix entry associated with s, a, m. We set $\mathbf{W}(s, a, m) = 2$ for all m if a is chosen at s by *CVaR VI Expected MDP*, and $\mathbf{W}(s, a, m) = 1$ for all m otherwise.

E Additional Results

E.1 Comparison with Random Action Expansion

Table 2 includes results for *RA-BAMCP* when the actions for the adversary are expanded randomly, rather than using Bayesian optimisation. For these results with random action expansions, we used double the number of trials that were used with Bayesian optimisation: 200,000 trials at the root node, and 50,000 trials per step thereafter. This means that the total computational resources used with and without Bayesian optimisation was comparable. Even with a comparable amount of computation time, *RA-BAMCP* with random action expansions performed worse than *RA-BAMCP* using Bayesian optimisation to select actions to expand.

Method	Time (s)	$\text{CVaR}_{0.03}$	$\text{CVaR}_{0.2}$	Expected Value
RA-BAMCP (Random expansions, $\alpha = 0.03$)	29.2 (0.1)	8.24 (0.29)	9.94 (0.06)	10.73 (0.05)
RA-BAMCP (Random expansions, $\alpha = 0.2$)	72.6 (0.1)	0.0 (0.0)	18.39 (0.99)	54.29 (0.53)

(a) Betting Game domain

Method	Time (s)	$\text{CVaR}_{0.03}$	$\text{CVaR}_{0.2}$	Expected Value
RA-BAMCP (Random expansions, $\alpha = 0.03$)	214.7 (0.1)	16.06 (0.35)	24.63 (0.25)	36.92 (0.20)
RA-BAMCP (Random expansions, $\alpha = 0.2$)	221.5 (0.2)	12.05 (0.46)	26.84 (0.38)	43.15 (0.26)

(b) Autonomous Car Navigation domain

Table 2: Results from evaluating the returns over 2000 episodes for *RA-BAMCP* using random action expansions for the adversary. Brackets indicate standard error of the mean. Time indicates average total computation time per episode to perform simulations.

E.2 Return Distribution in Betting Game

The histograms in Figure 3 show the return distributions for *RA-BAMCP* and *BAMCP* in the Betting Game domain.

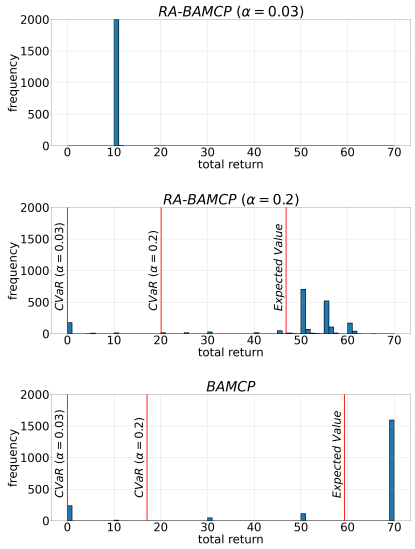


Figure 3: Histogram of returns received in the Betting Game domain.

E.3 Hyperparameter Tuning for Policy Gradient Method

Table 3 presents results for *CVaR BAMDP PG* after 2×10^6 training simulations using a range of learning rates. A learning rate of 0.001 performed the best across both domains and α values that we tested on. A learning rate of $\lambda = 0.01$ performed worse on the betting game with $\alpha = 0.2$, and a learning rate of $\lambda = 0.0001$ performed worse across both domains and confidence levels. Therefore, the results presented in the main body of the paper use a learning rate of $\lambda = 0.001$. Training curves using this learning rate can be found in Section E.4.

Method	$\text{CVaR}_{0.03}$	$\text{CVaR}_{0.2}$	Expected Value
CVaR BAMDP PG ($\lambda = 0.01$, $\alpha = 0.03$)	3.00 (0.0)	10.86 (0.33)	27.31 (0.24)
CVaR BAMDP PG ($\lambda = 0.001$, $\alpha = 0.03$)	2.90 (0.06)	10.57 (0.33)	27.34 (0.24)
CVaR BAMDP PG ($\lambda = 0.0001$, $\alpha = 0.03$)	2.31 (0.14)	8.74 (0.28)	26.19 (0.25)
CVaR BAMDP PG ($\lambda = 0.01$, $\alpha = 0.2$)	0.0 (0.0)	18.03 (0.91)	35.73 (0.27)
CVaR BAMDP PG ($\lambda = 0.001$, $\alpha = 0.2$)	0.00 (0.0)	19.85 (0.97)	46.65 (0.37)
CVaR BAMDP PG ($\lambda = 0.0001$, $\alpha = 0.2$)	0.0 (0.0)	14.72 (0.76)	43.14 (0.53)

(a) Betting Game domain

Method	$\text{CVaR}_{0.03}$	$\text{CVaR}_{0.2}$	Expected Value
CVaR BAMDP PG (Learning rate= 0.01, $\alpha = 0.03$)	24.0 (0.0)	25.02 (0.04)	28.97 (0.05)
CVaR BAMDP PG (Learning rate= 0.001, $\alpha = 0.03$)	23.92 (0.08)	25.38 (0.05)	28.97 (0.05)
CVaR BAMDP PG (Learning rate= 0.0001, $\alpha = 0.03$)	20.11 (2.12)	24.86 (0.34)	29.00 (0.09)
CVaR BAMDP PG (Learning rate= 0.01, $\alpha = 0.2$)	3.7 (0.97)	24.88 (0.60)	51.78 (0.36)
CVaR BAMDP PG (Learning rate= 0.001, $\alpha = 0.2$)	10.05 (0.74)	24.70 (0.39)	49.67 (0.36)
CVaR BAMDP PG (Learning rate= 0.0001, $\alpha = 0.2$)	-13.25 (2.78)	19.94 (0.91)	51.19 (0.42)

(b) Autonomous Car Navigation domain

Table 3: Results from evaluating the returns over 2000 episodes for *CVaR BAMDP PG* after training for 2×10^6 simulations using different learning rates. Brackets indicate standard error of the mean.

E.4 Training Curves for Policy Gradient Method

The training curves presented Figures 4-7 illustrate the performance of the policy throughout training using the policy gradient approach, with the learning rate set to $\lambda = 0.001$. To generate the curves, after every 20,000 training simulations the policy is executed for 2,000 episodes and the CVaR is evaluated based on these episodes.

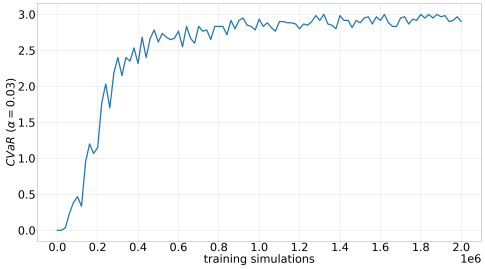


Figure 4: Training curve for policy gradient optimisation in the betting game domain with $\alpha = 0.03$.

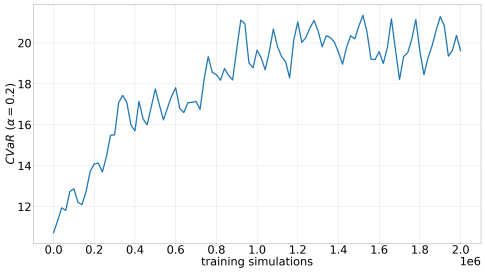


Figure 5: Training curve for policy gradient optimisation in the betting game domain with $\alpha = 0.2$.

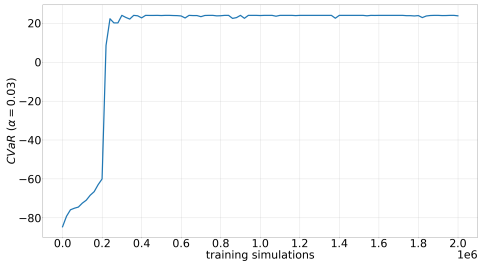


Figure 6: Training curve for policy gradient optimisation in the navigation domain with $\alpha = 0.03$.

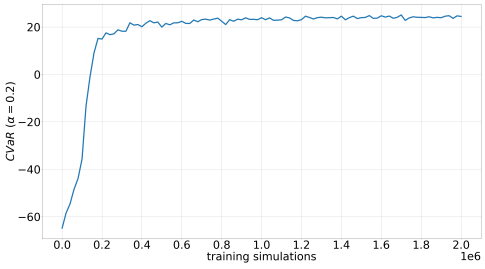


Figure 7: Training curve for policy gradient optimisation in the navigation domain with $\alpha = 0.2$.

Appendices of Chapter 6

Appendices

A Proof of Proposition 2

Proposition 2 (Model Gradient)**.** *Let ϕ denote the parameters of a parametric MDP model $\widehat{T}_\phi$, and let V_ϕ^π denote the value function for policy π in $\widehat{T}_\phi$. Then:*

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi, (s',r)\sim\widehat{T}_\phi}\big[(r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi \log \widehat{T}_\phi(s',r|s,a)\big] \tag{4}$$

Proof: The proof is analogous to the proof of the policy gradient theorem [61, 62]. We start by expressing the value function using Bellman's equation, for some state s:

$$\begin{aligned} V_\phi^\pi(s) &= \sum_a \pi(a|s) Q_\pi(s,a) \\ &= \sum_a \pi(a|s) \sum_{s',r} (r+\gamma V_\phi^\pi(s'))\cdot \widehat{T}_\phi(s',r|s,a). \end{aligned} \tag{10}$$

Applying the product rule:

$$\begin{aligned} \nabla_\phi V_\phi^\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} \Big[(r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi \widehat{T}_\phi(s',r|s,a) + \widehat{T}_\phi(s',r|s,a)\cdot\nabla_\phi(r+\gamma V_\phi^\pi(s'))\Big] \\ &= \sum_a \pi(a|s)\sum_{s',r}(r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi\widehat{T}_\phi(s',r|s,a) + \gamma\sum_a\pi(a|s)\sum_{s'}\widehat{T}_\phi(s'|s,a)\cdot\nabla_\phi V_\phi^\pi(s') \end{aligned} \tag{11}$$

Define $\psi(s) := \sum_a \pi(a|s)\sum_{s',r}(r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi\widehat{T}_\phi(s',r|s,a)$, and additionally define $\rho_\phi^\pi(s\to x, n)$ as the probability of transitioning from state s to x by executing π for n steps in the MDP model defined by $\widehat{T}_\phi$. Using these definitions, we can rewrite the above equation as:

$$\begin{aligned} \nabla_\phi V_\phi^\pi(s) &= \psi(s) + \gamma\sum_{s'}\rho_\phi^\pi(s\to s',1)\cdot\nabla_\phi V_\phi^\pi(s') \\ &= \psi(s) + \gamma\sum_{s'}\rho_\phi^\pi(s\to s',1)\big[\psi(s') + \gamma\sum_{s''}\rho_\phi^\pi(s'\to s'',1)\cdot\nabla_\phi V_\phi^\pi(s'')\big] \\ &= \psi(s) + \gamma\sum_{s'}\rho_\phi^\pi(s\to s',1)\psi(s') + \gamma^2\sum_{s''}\rho_\phi^\pi(s\to s'',2)\cdot\nabla_\phi V_\phi^\pi(s'') \\ &= \sum_{s'}\sum_{t=0}^{\infty}\gamma^t\rho_\phi^\pi(s\to s',t)\psi(s') \qquad \text{(continuing to unroll)} \end{aligned} \tag{12}$$

We have that $V_\phi^\pi = \sum_s \mu_0(s)\cdot V_\phi^\pi(s)$ by definition. Therefore,

$$\begin{aligned} \nabla_\phi V_\phi^\pi &= \nabla_\phi\big(\sum_s \mu_0(s)\cdot V_\phi^\pi(s)\big) \\ &= \sum_s \mu_0(s)\cdot\nabla_\phi V_\phi^\pi(s) \\ &= \sum_s \mu_0(s)\sum_{s'}\sum_{t=0}^{\infty}\gamma^t\rho_\phi^\pi(s\to s',t)\psi(s') \\ &= \sum_s d_\phi^\pi(s)\psi(s) \end{aligned} \tag{13}$$

where $d^\pi_\phi(s)$ is the (improper) discounted state visitation distribution of the policy π in the MDP model $\widehat{T}_\phi$, under initial state distribution μ_0. Finally, we apply the log-derivative trick to arrive at the final result:

$$\begin{aligned}
\nabla_\phi V^\pi_\phi &= \sum_s d^\pi_\phi(s) \sum_a \pi(a|s) \sum_{s',r} (r + \gamma V^\pi_\phi(s')) \cdot \nabla_\phi \widehat{T}_\phi(s', r|s, a) \\
&= \sum_s d^\pi_\phi(s) \sum_a \pi(a|s) \sum_{s',r} \widehat{T}_\phi(s', r|s, a)(r + \gamma V^\pi_\phi(s')) \frac{\nabla_\phi \widehat{T}_\phi(s', r|s, a)}{\widehat{T}_\phi(s', r|s, a)} \\
&= \mathbb{E}_{s\sim d^\pi_\phi, a\sim\pi, (s',r)\sim\widehat{T}_\phi} \left[(r + \gamma V^\pi_\phi(s')) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \right] \quad \square
\end{aligned} \tag{14}$$

A.1 Model Gradient with State-Action Value Baseline

We can subtract $Q^\pi_\phi(s, a)$ as a baseline without biasing the gradient estimate:

$$\nabla_\phi V^\pi_\phi = \mathbb{E}_{s\sim d^\pi_\phi, a\sim\pi, (s',r)\sim\widehat{T}_\phi} \left[(r + \gamma V^\pi_\phi(s') - Q^\pi_\phi(s, a)) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \right] \tag{15}$$

Proof: We can subtract any quantity which does not depend on s' or r without changing the value of the expression because the subtracted quantity is zero. Specifically, for the baseline of $Q^\pi_\phi(s, a)$ we have that:

$$\begin{aligned}
&\mathbb{E}_{s\sim d^\pi_\phi, a\sim\pi, (s',r)\sim\widehat{T}_\phi} \left[Q^\pi_\phi(s, a) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \right] \\
&\quad = \sum_s d^\pi_\phi(s) \sum_a \pi(a|s) \sum_{s',r} \widehat{T}_\phi(s', r|s, a) \cdot Q(s, a) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \\
&\quad = \sum_s d^\pi_\phi(s) \sum_a \pi(a|s) \cdot Q(s, a) \cdot \sum_{s',r} \nabla_\phi \widehat{T}_\phi(s', r|s, a) \\
&\quad = \sum_s d^\pi_\phi(s) \sum_a \pi(a|s) \cdot Q(s, a) \cdot \nabla_\phi \sum_{s',r} \widehat{T}_\phi(s', r|s, a) \\
&\quad = \sum_s d^\pi_\phi(s) \sum_a \pi(a|s) \cdot Q(s, a) \cdot \nabla_\phi(1) \\
&\quad = 0
\end{aligned} \tag{16}$$

B Implementation Details

The code for RAMBO is available at github.com/marc-rigter/rambo. Our implementation builds upon the official MOPO codebase.

B.1 Model Training

We represent the model as an ensemble of neural networks that output a Gaussian distribution over the next state and reward given the current state and action:

$$\widehat{T}_\phi(s', r|s, a) = \mathcal{N}(\mu_\phi(s, a), \Sigma_\phi(s, a)).$$

Following previous works [75, 76], during the initial maximum likelihood model training (Line 1 of Algorithm 1) we train an ensemble of 7 such dynamics models and pick the best 5 models based on the validation error on a held-out test set of 1000 transitions from the dataset $\mathcal{D}$. Each model in the ensemble is a 4-layer feedforward neural network with 200 hidden units per layer.

After the maximum likelihood training, RAMBO updates all of these 5 models adversarially according to Algorithm 1. For each model rollout, we randomly pick one of the 5 dynamics models to simulate

Table 4: Base hyperparameter configuration which is shared across all runs of RAMBO.

	Hyperparameter	Value
SAC	critic learning rate	3e-4
	actor learning rate	1e-4
	discount factor (γ)	0.99
	soft update parameter (τ)	5e-3
	target entropy	-dim(A)
	batch size	256
MBPO	no. of model networks	7
	no. of elites	5
	model learning rate	3e-4
	ratio of real data (f)	0.5
	model training batch size	256
	number of iterations (n_{iter})	2000

the rollout. The learning rate for the model training is 3e-4 for both the MLE pretraining, and the adversarial training. The model is trained using the Adam optimiser [26].

For the MLE pretraining, the batch size is 256. For each adversarial update in Equation 9 we sample a batch of 256 transitions from $\mathcal{D}$ to compute the maximum likelihood term, and 256 synthetic transitions for the model gradient term. The hyperparameters used for model training are summarised in Table 4.

B.2 Policy Training

We sample a batch of 256 transitions to train the policy and value function using soft actor-critic (SAC) [19]. We set the ratio of real data to $f = 0.5$ for all datasets, meaning that we sample 50% of the batch transitions from $\mathcal{D}$ and the remaining 50% from $\mathcal{D}_{\widehat{T}_\phi}$. We chose this value because it was used for most datasets by COMBO [75] and we found that it worked well. We represent the Q-networks and the policy as three layer neural networks with 256 hidden units per layer.

For SAC [19] we use automatic entropy tuning, where the entropy target is set to the standard heuristic of $-\dim(A)$. The only hyperparameter that we modify from the standard implementation of SAC is the learning rate for the actor, which we set to 1e-4 as this was reported to work better in the CQL paper [31] which also utilised SAC. For reference, the hyperparameters used for SAC are included in Table 4.

B.3 RAMBO Implementation Details

For each iteration of RAMBO we perform 1000 gradient updates to the actor-critic algorithm, and 1000 adversarial gradient updates to the model. For each run, we perform 2000 iterations. The base hyperparameter configuration for RAMBO that is shared across all runs is shown in Table 4.

The only parameters that we vary between datasets are the length of the synthetic rollouts, k, the weighting of the adversarial term in the model update, λ, and the choice of rollout policy. Details are included in the Hyperparameter Tuning section below.

B.4 Hyperparameter Tuning

We found that the hyperparameters that have the largest influence on the performance of RAMBO are the length of the synthetic rollouts (k), and the weighting of the adversarial term in the model update (λ). For the rollout length, we found that a value of either 2 or 5 worked well across most datasets. This is a slight modification to the values of $k \in \{1, 5\}$ that are typically used in previous model-based offline RL algorithms [8, 75, 76].

To arrive at a suitable value for the adversarial weighting, we used the intuition that we must have $\lambda \ll 1$. This is necessary so that the MLE term dominates in the loss function in Equation 9 for in-distribution samples. This ensures that the model still fits the dataset accurately despite the adversarial

Table 5: Hyperparameters used by RAMBO and RAMBO$^{\text{OFF}}$ for each dataset. k is the rollout length, and λ is the adversary loss weighting. The hyperparameters for RAMBO are those with the best performance from the three configurations tested (see Table 6). The hyperparameters for RAMBO$^{\text{OFF}}$ are those which obtained the lowest value of the offline heuristic in Table 6. As indicated, a random rollout policy was used for some of the AntMaze domains as we found that this performed better.

		RAMBO		RAMBO$^{\text{OFF}}$		
		k	λ	k	λ	Rollout Policy
Random	HalfCheetah	5	0	2	3e-4	Current Policy
	Hopper	2	3e-4	5	0	
	Walker2D	5	0	5	3e-4	
Medium	HalfCheetah	5	3e-4	2	3e-4	
	Hopper	5	3e-4	5	3e-4	
	Walker2D	5	0	5	0	
Medium Replay	HalfCheetah	5	3e-4	5	0	
	Hopper	2	3e-4	2	3e-4	
	Walker2D	5	0	5	0	
Medium Expert	HalfCheetah	5	3e-4	2	3e-4	
	Hopper	5	3e-4	5	3e-4	
	Walker2D	2	3e-4	2	3e-4	
AntMaze	Umaze	5	3e-4	5	3e-4	Current Policy
	Medium-Play	5	0	5	3e-4	Uniform Random
	Large-Play	5	3e-4	5	3e-4	
	Umaze-Diverse	5	0	5	0	
	Medium-Diverse	5	0	2	3e-4	
	Large-Diverse	5	0	5	0	

training. We observed that if λ is set too high, the training can be unstable and the Q-function can become extremely negative. If λ is too small, the adversarial training has no effect as the model is updated too slowly. We found that a value of 3e-4 generally obtained good performance.

For all MuJoCo datasets we performed model rollouts using the current policy, but for some AntMaze datasets we used a random rollout policy as we found this performed better. The rollout policies used are listed in Table 5. For the MuJoCo datasets, we initialised the policy using behaviour cloning (BC) which is common practice in offline RL [25, 74]. Ablation results without the BC initialisation are in Appendix C.4. We did not use the BC initialisation for the AntMaze problems.

For the rollout length and the adversarial weighting, we tested the following three configurations on each dataset: $(k, \lambda) \in \{(2, \text{3e-4}), (5, \text{3e-4}), (5, 0)\}$. We included $(k, \lambda) = (5,\ 0)$ as we found that an adversarial weighting of 0 worked well for some datasets. As described in Section 6, to evaluate RAMBO we ran each of the three hyperparameter configurations for five seeds each, and reported the best performance across the three configurations. The results for the best configuration are in Table 1, and the corresponding final hyperparameters chosen can be found in Table 5. The results for each of the three configurations can be found in Table 6.

Offline selection of hyperparameters is an important topic in offline RL [47, 77]. This is because in some applications the selection of hyperparameters based on online performance may be infeasible. Therefore, we present additional results in Table 1 where we select between the three choices of hyperparameters offline using a simple heuristic based on the magnitude and stability of the Q-values during training. After selecting the hyperparameters using the heuristic, we ran a further five seeds using these parameters to produce the final result. We refer to this approach as RAMBO$^{\text{OFF}}$. The hyperparameters used by RAMBO$^{\text{OFF}}$ are also included in Table 5. Details of the heuristic used to select the hyperparameter configuration for RAMBO$^{\text{OFF}}$ can be found in the following subsection.

B.5 Offline Hyperparameter Selection Method

Deep RL algorithms are highly sensitive to the choice of hyperparameters [3]. There is no consensus on how parameters should be selected for offline RL [47]. Some possibilities include a) selection based on online evaluation [8, 9, 25, 39, 72, 76], b) selection based on an offline heuristic [47, 75], and c) methods based on off-policy evaluation [47, 77].

To address applications in which online evaluation is possible, we evaluated RAMBO using approach a) which is the most common practice among model-based offline RL algorithms [8, 25, 37, 39, 76]. To consider situations where online tuning is not possible, we also evaluate using approach b) to select the hyperparameters for each dataset using a simple heuristic in a similar vein to [75]. We refer to this second approach as RAMBO$^{\text{OFF}}$.

To arrive at a heuristic that can be evaluated offline, we use the intuition that the value function should be regularised (i.e. low) as well as stable during training. Therefore, our heuristic makes the final hyperparameter selection from the set of candidate parameters according to: $\min \left(Q_{\text{avg}} + Q_{\text{var}}\right)$, where Q_{avg} is the average Q-value at the end of training, and Q_{var} is the variance of the average Q-value over the final 100 iterations of training.

The values of this heuristic for each of the three configurations that we test are the values in the brackets provided in Table 6. The bolded values in the table indicate the lowest value for the heuristic, which is the parameter configuration chosen for RAMBO$^{\text{OFF}}$. After choosing the hyperparameters in this manner, we *rerun* the algorithm for a further five seeds per dataset. The performance for these additional runs are the results reported in Table 1 for RAMBO$^{\text{OFF}}$.

The results in Table 1 indicate that RAMBO$^{\text{OFF}}$ obtains strong performance compared to existing baselines on the MuJoCo domains. This indicates that it is possible to obtain solid performance with RAMBO by choosing the final hyperparameters according to the magnitude and stability of the Q-values. Unsurprisingly however, RAMBO performs slightly better than RAMBO$^{\text{OFF}}$. This shows that better performance is obtained using online hyperparameter tuning.

B.6 Evaluation Procedure

We report the average undiscounted normalized return averaged over the last 10 iterations of training, with 10 evaluation episodes per iteration. We evaluated the SAC policy by deterministically taking the mean action. The normalization procedure is that proposed by [14], where 100 represents an expert policy and 0 represents a random policy.

B.7 Baselines and Domains

We compare RAMBO against state of the art model-based (COMBO [75], MOReL [25], RepB-SDE [34] and MOPO [76]) and model-free (CQL [31], IQL [28], and TD3+BC [15]) offline RL algorithms on the D4RL benchmarks [14]. The D4RL benchmarks are released with the Apache License 2.0. To facilitate a fair comparison, we provide results for all algorithms for the MuJoCo-v2 D4RL datasets which contain more performant data than v0 [37].

Because the original COMBO, MOPO, and CQL papers use the MuJoCo-v0 datasets we report the results for these algorithms on the MuJoCo-v2 datasets from [67]. The MOReL, IQL, RepB-SDE, and TD3+BC papers include evaluations on the MuJoCo-v2 datasets, so for these algorithms we include the results from the original papers.

For AntMaze-v0, we report the results from the original papers for CQL, IQL, and TD3+BC. For COMBO, MOPO, RepB-SDE, and MOReL, the results are taken from [67]. Following the IQL paper [28], we subtract 1 from the rewards of the AntMaze datasets for our experiments.

B.8 Computational Resources

During our experiments, each run of RAMBO has access to 2 CPUs of an Intel Xeon Platinum 8259CL processor at 3.1GHz, and half of an Nvidia T4 GPU. With this hardware, each run of RAMBO takes 24-30 hours.

For our main evaluation of RAMBO, we ran five seeds for each of three parameter configurations for 18 tasks resulting in a total of 270 runs. This required approximately 150 GPU-days of compute.

C Additional Results

C.1 Single Transition Example

Description In this domain, the state and action spaces are one-dimensional. The agent executes a single action, $a \in [-1, 1]$, from initial state s_0. After executing the action, the agent transitions to s' and receives a reward equal to the successor state, i.e. $r(s') = s'$, and the episode terminates.

The actions in the dataset are sampled uniformly from $a \in [-0.75, 0.7] \cup [-0.15, -0.1] \cup [0.1, 0.15] \cup [0.7, 0.75]$. In the MDP for this domain, the transition distribution for successor states is as follows:

- $s' \sim \mathcal{N}(\mu = 1,\ \sigma = 0.2)$, for $a \in [-0.8,\ -0.65]$.
- $s' \sim \mathcal{N}(\mu = 0.5,\ \sigma = 0.2)$, for $a \in [-0.2,\ -0.05]$.
- $s' \sim \mathcal{N}(\mu = 1.25,\ \sigma = 0.2)$, for $a \in [0.05,\ 0.2]$.
- $s' \sim \mathcal{N}(\mu = 1.5,\ \sigma = 0.2)$, for $a \in [0.65,\ 0.8]$.
- $s' = 0.5$, for all other actions.

The transitions to successor states from the actions in the dataset are illustrated in Figure 1.

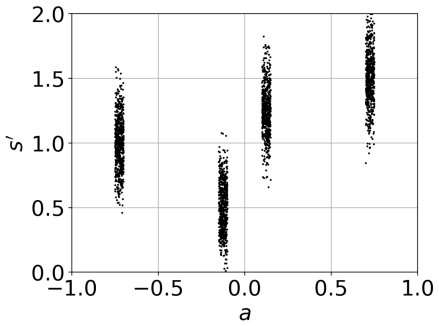


Figure 1: Transition data for the Single Transition Example after executing action a from s_0.

Comparison between RAMBO and COMBO To generate the comparison, we run both algorithms for 50 iterations. To choose the regularisation parameter for COMBO, we sweep over $\beta \in \{0.1, 0.2, 0.5, 1, 5\}$ and choose the parameter with the best performance. Note that 0.5, 1, and 5 are the values used in the original COMBO paper [75], so we consider lower values of regularisation than in the original paper. For RAMBO we use λ =3e-2. We used the implementation of COMBO from github.com/takuseno/d3rlpy.

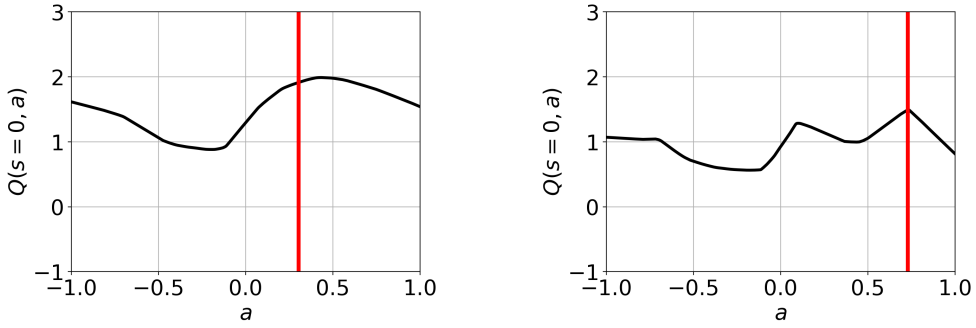
Figure 2 shows the Q-values produced by RAMBO throughout training. We can see that after 5 iterations, the Q-values for actions which are outside of the dataset are initially *optimistic*, and over-estimate the true values. However, after 50 iterations the Q-values for out of distribution actions have been regularised by RAMBO. The action selected by the policy after 50 iterations is the optimal action in the dataset, $a \in [0.7, 0.75]$. Thus, we see that RAMBO introduces pessimism *gradually* as the adversary is trained to modify the transitions to successor states.

For COMBO, the best performance averaged over 20 seeds was obtained for $\beta = 0.2$, and therefore we report results for this value of the regularisation parameter. Figure 3 illustrates the Q-values produced by COMBO throughout training (with $\beta = 0.2$). We see that at both 5 iterations and 50 iterations, the value estimates for actions outside of the dataset are highly pessimistic. In the run illustrated for COMBO, the action selected by the policy after 50 iterations is $a \in [0.1, 0.15]$, which is not the optimal action in the dataset. Figure 3 shows that the failure to find an optimal action is due to the gradient-based policy optimisation becoming stuck in a local maxima of the Q-function.

These results highlight a difference in the behaviour of RAMBO compared to COMBO. For RAMBO, pessimism is introduced gradually as the adversary is trained to modify the transitions to successor states. For COMBO, pessimism is introduced at the outset as it is part of the value function update

throughout the entirety of training. Additionally, the regularisation of the Q-values for out of distribution actions appears to be less aggressive for RAMBO than for COMBO.

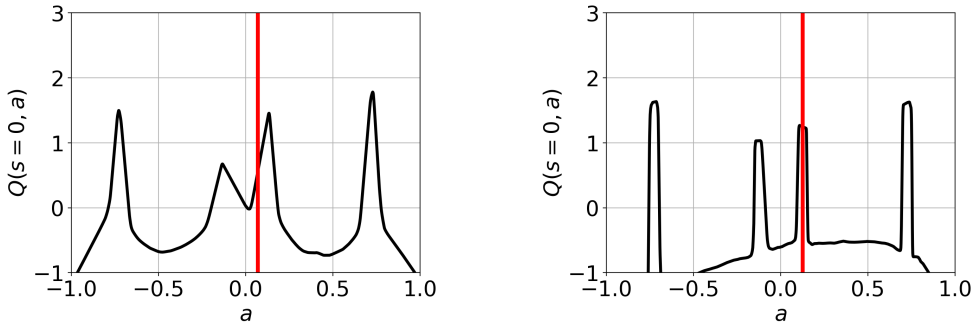
The results averaged over 20 seeds in Table 3 show that RAMBO consistently performs better than COMBO for this problem. This suggests that the gradual introduction of pessimism produced by RAMBO means that the policy optimisation procedure is less likely to get stuck in poor local maxima for this example. The downside of this behaviour is that it may take more iterations for RAMBO to find a performant policy. Modifying other algorithms to gradually introduce pessimism may be an interesting direction for future research.



(a) 5 iterations

(b) 50 iterations

Figure 2: Q-values at the initial state during the training of RAMBO on the Single Transition Example. The red line indicates the mean action of the policy.



(a) 5 iterations

(b) 50 iterations

Figure 3: Q-values at the initial state during the training of COMBO on the Single Transition Example ($\beta = 0.2$). The red line indicates the mean action of the policy.

C.2 Results for each Hyperparameter Configuration

In Table 6, we report the results for each of the three parameter configurations that we test. The highlighted values indicate the best performance for each dataset, and are the results reported for RAMBO in Table 1. In Table 6 we also report the values of the offline hyperparameter selection heuristic in brackets. The bold values indicate the lowest value of the heuristic, which is the hyperparameter configuration selected for RAMBO$^{\text{OFF}}$.

Table 6: Performance of RAMBO across each of the three configurations of the rollout length, k, and the adversarial weighting, λ, that we tested. Values indicate the average normalized return at the end of training. Each configuration was run for 5 seeds. $\pm$ captures the standard deviation over seeds. Highlighted values indicate the best performance for each dataset. The values in brackets indicate the value of the offline tuning heuristic described in Appendix B.5. "$-$" indicates that the value function diverged.

		$k = 2, \lambda = 3e-4$	$k = 5, \lambda = 3e-4$	$k = 5, \lambda = 0$
Random	HalfCheetah	34.5 ± 3.7 (**348.1**)	38.2 ± 3.9 (408.6)	40.0 ± 2.3 (377.6)
	Hopper	21.6 ± 8.0 (1179.7)	21.3 ± 9.2 (1293.2)	15.1 ± 9.4 (**188.9**)
	Walker2D	$-$	0.2 ± 0.5 (**1.2e8**)	11.5 ± 10.5 ($2.5e9$)
Medium	HalfCheetah	72.5 ± 4.4 (**671.1**)	77.6 ± 1.5 (717.4)	75.5 ± 6.0 (720.6)
	Hopper	$-$	92.8 ± 6.0 (**300.4**)	47.5 ± 36.0 (314.1)
	Walker2D	82.2 ± 11.4 (523.6)	76.8 ± 4.8 (2124.2)	86.9 ± 2.7 (**339.4**)
Medium Replay	HalfCheetah	65.4 ± 3.7 (623.5)	68.9 ± 2.3 (647.2)	64.7 ± 3.2 (**608.4**)
	Hopper	96.6 ± 7.0 (**262.4**)	93.5 ± 10.3 (309.1)	36.7 ± 9.5 (263.4)
	Walker2D	6.7 ± 2.5 (2.3e7)	62.1 ± 33.5 (4.3e6)	85.0 ± 15.0 (**259.0**)
Medium Expert	HalfCheetah	78.1 ± 6.6 (**987.1**)	93.7 ± 10.5 (1008.9)	92.9 ± 11.9 (1013.2)
	Hopper	36.4 ± 16.7 (344.1)	83.3 ± 9.1 (**342.8**)	77.6 ± 14.3 (344.4)
	Walker2D	68.3 ± 20.6 (**371.9**)	31.7 ± 49.8 ($2.0e7$)	50.8 ± 57.8 ($6.9e9$)
AntMaze	Umaze	8.8 ± 8.2 (-48.3)	25.0 ± 12.0 (**-54.2**)	3.8 ± 5.8 (-51.5)
	Medium-Play	5.8 ± 6.6 (1421)	8.4 ± 13.5 (**-70.77**)	16.4 ± 17.9 (-68.10)
	Large-Play	0.0 ± 0.0 (-77.9)	0.0 ± 0.0 (**-78.1**)	0.0 ± 0.0 (-77.9)
	Umaze-Diverse	0.0 ± 0.0 (790.8)	0.0 ± 0.0 (68.2)	0.0 ± 0.0 (**-65.4**)
	Medium-Diverse	3.8 ± 1.7 (**-72.4**)	1.6 ± 1.8 (-72.2)	23.2 ± 14.2 (-71.2)
	Large-Diverse	$-$	0.0 ± 0.0 (6.0e8)	2.4 ± 3.3 (**3.5e6**)

C.3 Visualisation of Adversarially Trained Dynamics Model

To visualise the influence of training the transition dynamics model in an adversarial manner as proposed by RAMBO, we consider the following simple example MDP. In the example, the state space (S) and action space (A) are 1-dimensional with, $A = [-1, 1]$. For this example, we assume that the reward function is known and is equal to the current state, i.e. $R(s, a) = s$, meaning that greater values of s have greater expected value. The true transition dynamics to the next state s' depend on the action but are the same regardless of the initial state, s.

In Figure 4 we plot the data present in the offline dataset for this MDP. Note that the actions in the dataset are sampled from a subset of the action space: $a \in [-0.3, 0.3]$. Because greater values of s' correspond to greater expected value, the transition data indicates that the optimal action covered by the dataset is $a = 0.3$.

In Figure 5 we plot the output of running naïve model-based policy optimisation (MBPO) on this illustrative offline RL example. Figure 5a illustrates the MLE transition function used by MBPO. The transition function fits the dataset for $a \in [-0.3, 0.3]$. Outside of the dataset, it predicts that an action of $a \approx 1$ transitions to the best successor state. Figure 5b shows that applying a reinforcement learning algorithm (SAC) to this model results in the value function being over-estimated, and $a \approx 1$ being predicted as the best action. This illustrates that the policy learns to exploit the inaccuracy in the model and choose an out-of-distribution action.

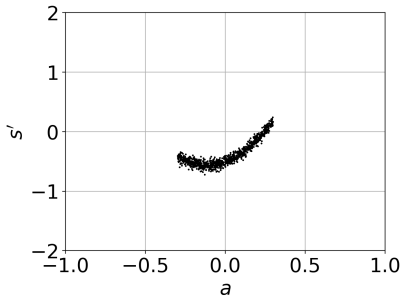
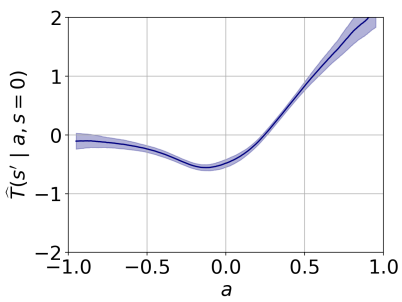
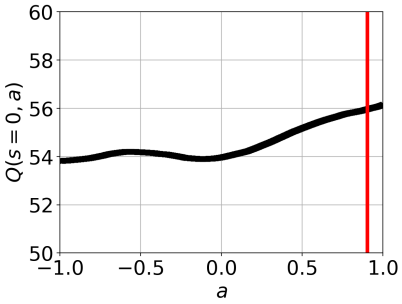


Figure 4: Transition data for illustrative example. The transition function is the same regardless of the initial state. The actions in the dataset are sampled from $a \in [-0.3, 0.3]$.

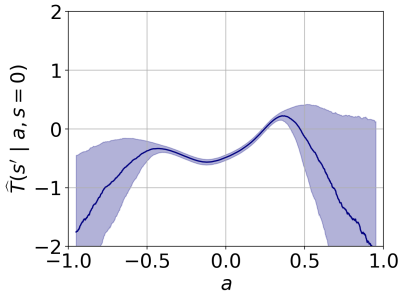


(a) Maximum likelihood estimate of the transition function, which is used by MBPO. Shaded area indicates ± 1 SD of samples generated by the ensemble.

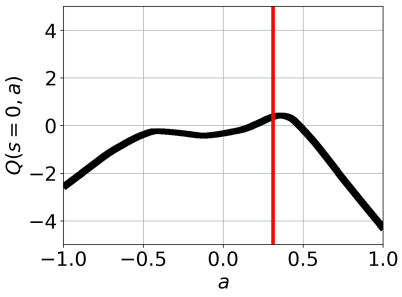


(b) Q-values after running SAC for 15 iterations. The values are significantly over-estimated. The red line indicates the mean action taken by the SAC policy.

Figure 5: Plots generated by running naïve MBPO on the illustrative MDP example.



(a) Model which is adversarially trained using RAMBO. Shaded area indicates ± 1 SD of samples generated by the ensemble. The transition function accurately predicts the dataset for in-distribution actions, but predicts transitions to low value states for actions which are out of distribution.



(b) Q-values from SAC critic. The red line indicates the mean action taken by the policy trained using SAC, which is $a \approx 0.3$ (the best in-distribution action). Training on pessimistic synthetic transitions regularises the Q-values for out-of-distribution actions.

Figure 6: Output generated by running RAMBO for 15 iterations on the illustrative MDP example, using an adversary weighting of λ =3e-2.

In Figure 6 we show plots generated after running RAMBO on this example for 15 iterations, using an adversary weighting of $\lambda =$ 3e-2. Figure 6a shows that the transition function still accurately predicts the transitions within the dataset. This is because choosing $\lambda \ll 1$ means that the MLE loss dominates for state-action pairs within the dataset. Due to the adversarial training, for actions $a \notin [-0.3, 0.3]$ which are out of the dataset distribution, the transition function generates pessimistic transitions to low values of s, which have low expected value.

As a result, low values are predicted for out-of-distribution actions, and the SAC agent illustrated in 6b learns to take action $a \approx 0.3$, which is the best action which is within the distribution of the dataset. This demonstrates that by generating pessimistic synthetic transitions for out-of-distribution actions, RAMBO enforces conservatism and prevents the learnt policy from taking state-action pairs which have not been observed in the dataset.

C.4 Ablation of Behaviour Cloning Initialisation

For the MuJoCo locomotion experiments we initialised the policy using behaviour cloning, as noted in Section 6 and Appendix B.4. This is a common initialisation step used in previous works [25, 74]. In Table 7 we present ablation results where the behaviour cloning initialisation is removed, and the policy is initialised randomly. Table 7 indicates that the behaviour cloning initialisation results in a small improvement in the performance of RAMBO.

Table 7: Ablation of RAMBO with no behaviour cloning initialisation on the D4RL benchmark [14]. We report the normalised performance during the last 10 iterations of training averaged over 5 seeds.

		RAMBO	**RAMBO, No BC**
Random	HalfCheetah	40.0 ± 2.3	39.5 ± 3.5
	Hopper	21.6 ± 8.0	25.4 ± 7.5
	Walker2D	11.5 ± 10.5	0.0 ± 0.3
Medium	HalfCheetah	77.6 ± 1.5	77.9 ± 4.0
	Hopper	92.8 ± 6.0	87.0 ± 15.4
	Walker2D	86.9 ± 2.7	84.9 ± 2.6
Medium Replay	HalfCheetah	68.9 ± 2.3	68.7 ± 5.3
	Hopper	96.6 ± 7.0	99.5 ± 4.8
	Walker2D	85.0 ± 15.0	89.2 ± 6.7
Medium Expert	HalfCheetah	93.7 ± 10.5	95.4 ± 5.4
	Hopper	83.3 ± 9.1	88.2 ± 20.5
	Walker2D	68.3 ± 20.6	56.7 ± 39.0
MuJoCo-v2 Total:		826.2 ± 33.8	812.4 ± 47.8

Appendices of Chapter 7

Appendices

A. Risk Measures

In our experiments, we make use of two different risk measures in our algorithm: the conditional value at risk (CVaR) (Rockafellar & Uryasev, 2002), and the Wang risk measure (Wang, 2000). For parameter α, CVaR is the mean of the α-portion of the worst outcomes: $\rho_{\text{CVaR}}(Z, \alpha) = \mathbb{E}[z \sim Z \mid z \leq F_Z^{-1}(\alpha)]$, where F_Z is the CDF of Z. For parameter η, the Wang risk measure for random variable Z can be defined as the expectation under a distorted cumulative distribution function $\rho_{\text{Wang}}(Z, \eta) = \int_{-\infty}^{\infty} z \frac{\partial}{\partial z} g(F_Z(z)) \mathrm{d}z$, and the distortion function is $g(\tau) = \Phi(\Phi^{-1}(\tau) + \eta)$, where Φ is the CDF of the standard normal distribution. In both instances, the parameters α and η control the level of risk-aversion.

We test our approach with these two risk measures because CVaR is commonly used and easy to interpret, but only considers the α-portion of the distribution, while the Wang risk measure takes into consideration the entire distribution. However, the general approach we propose is compatible with any coherent risk measure.

A.1. Computing Perturbed Distributions

Here, we explain how the distribution over candidate successor states, $s' \in \Psi$ is modified for CVaR and the Wang risk measures in Algorithm 1. To avoid the need for excessive notation we assume that all sampled candidate successor states in Ψ are unique, and that no two successor states have exactly the same value, i.e. $V(s'_i) \neq V(s'_j)$, for any two different $s'_i, s'_j \in \Psi$.

Conditional Value at Risk Consider a probability space $(\Omega, \mathcal{F}, P)$, where Ω is the sample space, $\mathcal{F}$ is the event space, and P is a probability distribution over $\mathcal{F}$. For a continuous random variable Z, CVaR is the mean of the α-portion of the worst outcomes: $\rho_{\text{CVaR}}(Z, \alpha) = \mathbb{E}\big[z \sim Z \mid z \leq F_Z^{-1}(\alpha)\big]$, where F_Z is the CDF of Z. For CVaR at confidence level α, the corresponding risk envelope is:

$$\mathcal{B}_{\text{CVaR}_\alpha}(P) = \left\{\xi : \xi(\omega) \in \left[0, \frac{1}{\alpha}\right] \forall \omega, \int_{\omega \in \Omega} \xi(\omega) P(\omega) \mathrm{d}\omega = 1\right\} \tag{9}$$

The risk envelope allows the likelihood for any outcome to be increased by at most $1/\alpha$. In Algorithm 1, for state-action pair (s, a), we approximate the distribution over successor states as a discrete uniform distribution over candidate successor states $s' \in \Psi$, where Ψ is a set of m states sampled from $\overline{T}(s, a, s')$. This discrete distribution is $\widehat{T}(s, a, s') = \frac{1}{m}$ for all $s' \in \Psi$.

In Line 10 of Algorithm 1, we denote by $\widehat{\xi}^*$ the adversarial perturbation to the discrete transition distribution:

$$\widehat{\xi}^* = \underset{\xi \in \mathcal{B}_\rho(\widehat{T}(s,a,\cdot))}{\arg\min} \sum_{s' \in \Psi} \widehat{T}(s, a, s') \cdot \xi(s') \cdot V^\pi(s')$$

Define $\mathcal{V}_{s'}$ as the random variable representing the value of the successor state, $V^\pi(s')$, when the successor state is drawn from $s' \sim \widehat{T}(s, a, \cdot)$. Also define the value at risk at confidence level $\alpha \in (0, 1]$, which is the α-quantile of a distribution: $\text{VaR}_\alpha(Z) = \min\{z \mid F_Z(z) \geq \alpha\}$. Then, from the risk envelope in Equation 9, we can derive that the adversarial perturbation is:

$$\widehat{\xi}_{\text{CVaR}}(s') \cdot \widehat{T}(s, a, s') = \begin{cases} \frac{1}{m \cdot \alpha}, & \text{if } V(s') < \text{VaR}_\alpha(\mathcal{V}_{s'}). \\ 0, & \text{if } V(s') > \text{VaR}_\alpha(\mathcal{V}_{s'}). \\ 1 - \sum_{s' \in \Psi} \frac{1}{m \cdot \alpha} \mathbb{1}[V(s') < \text{Var}_\alpha], & \text{if } V(s') = \text{VaR}_\alpha(\mathcal{V}_{s'}). \end{cases} \tag{10}$$

The last line of Equation 10 ensures that the perturbed distribution is a valid probability distribution (as required by the risk envelope in Equation 9).

Wang Risk Measure For parameter η, the Wang risk measure for random variable, Z, can be defined as the expectation under a distorted cumulative distribution function $\rho_{\text{Wang}}(Z, \eta) = \int_{-\infty}^{\infty} z \frac{\partial}{\partial z} g(F_Z(z)) \mathrm{d}z$. The distortion function is $g(\tau) = \Phi(\Phi^{-1}(\tau) + \eta)$, where Φ is the CDF of the standard normal distribution.

To compute the perturbation to $\widehat{T}(s,a,\cdot)$ according to the Wang risk measure, we order the candidate successor states according to their value. We use the subscript to indicate the ordering of s'_i , where $V(s'_1) < V(s'_2) < V(s'_3) \ldots < V(s'_m)$. Then, the perturbed probability distribution is:

$$\widehat{\xi}_{\text{Wang}}(s'_i) \cdot \widehat{T}(s,a,s'_i) = \begin{cases} g(\frac{1}{m}), & \text{if } i = 1. \\ g(\frac{i}{m}) - g(\frac{i-1}{m}), & \text{if } 1 < i < m. \\ 1 - g(\frac{i-1}{m}), & \text{if } i = m. \end{cases} \tag{11}$$

The transition distribution perturbed according to the Wang risk measure assigns gradually decreasing probability to higher-value successor states. The parameter η controls the extent to which the probabilities are modified. For large $\eta >> 0$, almost all of the probability is placed on the worst outcome(s), and as $\eta \to 0$, the perturbed distribution approaches a uniform distribution.

B. Implementation Details

The code for the experiments can be found at github.com/marc-rigter/1R2R.

B.1. Model Training

We represent the model as an ensemble of neural networks that output a Gaussian distribution over the next state and reward given the current state and action:

$$\widehat{T}_\phi(s', r|s, a) = \mathcal{N}(\mu_\phi(s,a), \Sigma_\phi(s,a)).$$

Following previous works (Yu et al., 2021; 2020; Rigter et al., 2022b), during model training we train an ensemble of 7 such dynamics models and pick the best 5 models based on the validation error on a held-out test set of 1000 transitions from the dataset $\mathcal{D}$. Each model in the ensemble is a 4-layer feedforward neural network with 200 hidden units per layer. $P(T \mid \mathcal{D})$ is assumed to be a uniform distribution over the 5 models. Increasing the number of networks in the ensemble has been shown to improve uncertainty estimation (Lu et al., 2022). However, we used the same ensemble as previous works to facilitate a fair comparison.

The learning rate for the model training is 3e-4 for both the MLE pretraining, and it is trained using the Adam optimiser (Kingma & Ba, 2015). The hyperparameters used for model training are summarised in Table 4.

B.2. Policy Training

We sample a batch of 256 transitions to train the policy and value function using soft actor-critic (SAC) (Haarnoja et al., 2018). We set the ratio of real data to $f = 0.5$ for all datasets, meaning that we sample 50% of the batch transitions from $\mathcal{D}$ and the remaining 50% from $\widehat{\mathcal{D}}$. We chose this value because we found it worked better than using 100% synthetic data (i.e. $f = 0$), and it was found to work well in previous works (Yu et al., 2021; Rigter et al., 2022b). We represent the Q-networks and the policy as three layer neural networks with 256 hidden units per layer.

For SAC (Haarnoja et al., 2018) we use automatic entropy tuning, where the entropy target is set to the standard heuristic of $-\dim(A)$. The only hyperparameter that we modify from the standard implementation of SAC is the learning rate for the actor, which we set to 1e-4 as this was reported to work better in the CQL paper (Kumar et al., 2020) which also utilised SAC. For reference, the hyperparameters used for SAC are included in Table 4.

B.3. 1R2R Implementation Details

When sampling successor states in Algorithm 1, we sample $m = 10$ candidate successor states to generate the set Ψ. For each run, we perform 2,000 iterations of 1,000 gradient updates. The rollout policy used to generate synthetic data is the current policy. The base hyperparameter configuration for 1R2R that is shared across all runs is shown in Table 4.

In our main experiments, the only parameters that we vary between datasets are the length of the synthetic rollouts (k), and the risk measure parameter. Details are included in the Hyperparameter Tuning section below. In Appendix C.3, we additionally include results where we increase m from its base value, and increase the size of the ensemble.

B.4. 1R2R Hyperparameter Tuning

The hyperparameters that we tune for 1R2R are the length of the synthetic rollouts (k), and the parameter associated with the chosen risk measure (α for 1R2R$_{\text{CVaR}}$ or η for 1R2R$_{\text{Wang}}$). For each dataset, we select the parameters that obtain the best online performance for the desired objective for that dataset. The objectives for each of the datasets are: optimising expected value for D4RL MuJoCo; optimising static CVaR$_{0.1}$ for Stochastic MuJoCo; and optimising static CVaR$_{0.02}$ for HIVTreatment.

The rollout length is tuned from $k \in \{1, 5\}$. For each of the two versions of our algorithm, 1R2R$_{\text{CVaR}}$ and 1R2R$_{\text{Wang}}$, the parameters are tuned from the following settings:

- 1R2R$_{\text{CVaR}}$: the rollout length, k, and CVaR parameter, α, are selected from: $(k, \alpha) \in \{1, 5\} \times \{0.5, 0.7, 0.9\}$.
- 1R2R$_{\text{Wang}}$: the rollout length, k, and Wang risk parameter, η, are selected from: $(k, \eta) \in \{1, 5\} \times \{0.1, 0.5, 0.75\}$.

Note that while the risk measures are only moderately risk-averse, in the dynamic risk approach of 1R2R, they are applied recursively at each step. This can result in a high degree of risk-aversion over many steps (Gagne & Dayan, 2021).

Please refer to Table 5 for a list of the hyperparameters used for 1R2R for each dataset, and to Table 4 for the base set of hyperparameters that is shared across all datasets.

Table 4: Base hyperparameter configuration shared across all runs of 1R2R.

	Hyperparameter	Value
SAC	critic learning rate	3e-4
	actor learning rate	1e-4
	discount factor (γ)	0.99
	soft update parameter (τ)	5e-3
	target entropy	-dim(A)
	batch size	256
1R2R	no. of model networks	7
	no. of elites	5
	ratio of real data (f)	0.5
	no. of iterations (N_{iter})	2000
	no. of candidate successor states (m)	10
	rollout policy	current π

Table 5: Hyperparameters used by 1R2R for each dataset, chosen according to best online performance. k is the rollout length, α is the parameter for CVaR, and η is the parameter for the Wang risk measure.

		1R2R (CVaR)		1R2R (Wang)	
		k	α	k	η
Random	HalfCheetah	5	0.9	5	0.1
	Hopper	5	0.7	5	0.5
	Walker2D	5	0.9	5	0.1
Medium	HalfCheetah	5	0.9	5	0.1
	Hopper	5	0.9	5	0.1
	Walker2D	5	0.9	5	0.1
Medium Replay	HalfCheetah	5	0.9	5	0.1
	Hopper	1	0.5	1	0.75
	Walker2D	1	0.9	1	0.1
Medium Expert	HalfCheetah	5	0.9	5	0.1
	Hopper	5	0.9	5	0.1
	Walker2D	1	0.7	1	0.5
Medium	Hopper (Mod)	5	0.9	5	0.1
	Walker2D (Mod)	1	0.9	1	0.5
	Hopper (High)	5	0.9	5	0.1
	Walker2D (High)	1	0.7	1	0.1
Medium Replay	Hopper (Mod)	5	0.7	5	0.5
	Walker2D (Mod)	1	0.9	1	0.1
	Hopper (High)	5	0.5	5	0.75
	Walker2D (High)	1	0.9	1	0.1
Medium Expert	Hopper (Mod)	5	0.7	5	0.75
	Walker2D (Mod)	1	0.5	1	0.75
	Hopper (High)	5	0.9	5	0.5
	Walker2D (High)	1	0.7	1	0.5
	HIVTreatment	1	0.7	1	0.5

C. Additional Results

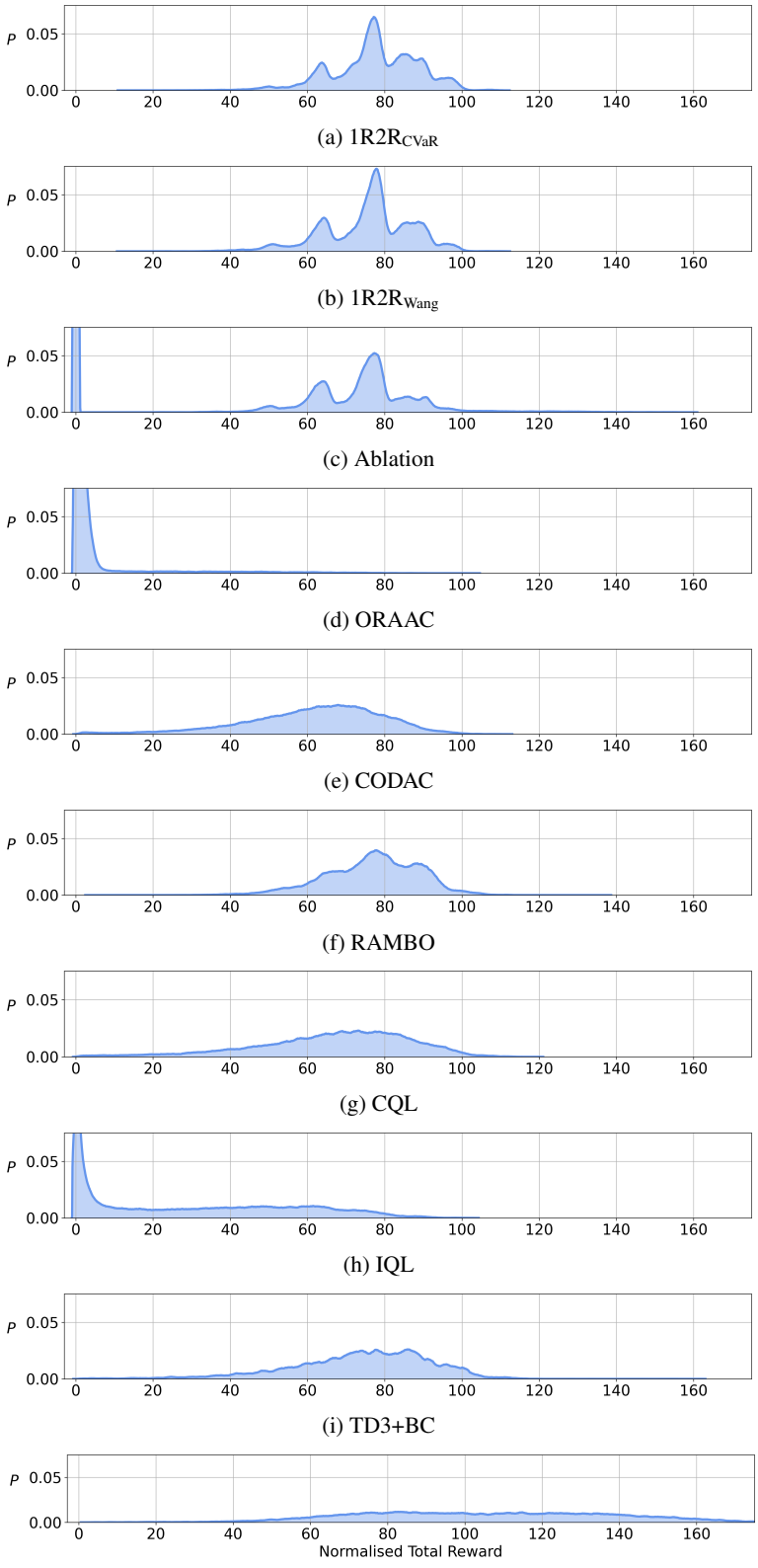
C.1. MuJoCo Stochastic Expected Value Results

Table 6: Expected value results for the Stochastic MuJoCo benchmarks using the normalisation procedure proposed by Fu et al. (2020). We report the normalised performance during the last 10 iterations of training averaged over 5 seeds. $\pm$ captures the standard deviation over seeds. Highlighted numbers indicate results within 10% of the best score. "div." indicates that the value function diverged for at least one run.

		1R2R$_{\text{CVaR}}$	1R2R$_{\text{Wang}}$	Ablation	O-RAAC	CODAC	RAMBO	CQL	IQL	TD3+BC
Medium	Hopper (Mod)	67.2 ± 5.7	65.9 ± 7.0	64.1 ± 10.0	15.6 ± 9.4	50.7 ± 3.0	72.9 ± 14.7	50.2 ± 0.6	50.6 ± 4.1	48.4 ± 1.4
	Walker2D (Mod)	67.3 ± 7.8	62.2 ± 6.3	27.6 ± 23.4	38.8 ± 7.1	44.1 ± 1.5	34.5 ± 2.2	58.4 ± 1.6	45.6 ± 1.3	41.4 ± 17.4
	Hopper (High)	57.1 ± 4.5	58.1 ± 5.9	53.0 ± 4.7	28.8 ± 11.0	69.9 ± 2.1	54.4 ± 4.4	74.7 ± 5.1	61.6 ± 8.0	66.3 ± 1.3
	Walker2D (High)	59.5 ± 11.8	58.8 ± 8.9	div.	41.6 ± 9.6	65.7 ± 0.6	56.9 ± 3.5	75.2 ± 1.3	54.8 ± 2.0	77.6 ± 2.3
Medium Replay	Hopper (Mod)	83.6 ± 18.6	76.8 ± 9.9	64.5 ± 9.8	15.8 ± 3.0	61.2 ± 10.7	67.6 ± 11.3	66.9 ± 10.8	-0.5 ± 0.0	38.2 ± 4.9
	Walker2D (Mod)	53.9 ± 9.4	51.7 ± 11.4	57.4 ± 3.0	24.2 ± 6.8	47.8 ± 19.7	63.6 ± 4.5	50.1 ± 1.9	16.2 ± 2.8	48.4 ± 1.5
	Hopper (High)	114.3 ± 10.8	102.0 ± 9.1	48.4 ± 16.0	26.2 ± 16.0	54.5 ± 2.9	54.1 ± 19.7	96.7 ± 13.0	15.1 ± 9.6	58.4 ± 6.1
	Walker2D (High)	84.1 ± 5.3	76.4 ± 6.0	75.4 ± 8.0	19.1 ± 4.7	80.4 ± 11.6	84.1 ± 4.0	61.1 ± 3.0	-0.1 ± 0.8	73.4 ± 2.3
Medium Expert	Hopper (Mod)	94.1 ± 17.2	71.3 ± 42.2	64.0 ± 41.7	32.8 ± 8.7	96.6 ± 8.8	71.9 ± 20.7	106.7 ± 3.8	107.5 ± 7.7	100.9 ± 2.0
	Walker2D (Mod)	72.2 ± 7.3	77.3 ± 9.1	div.	27.6 ± 12.4	81.7 ± 4.8	58.0 ± 22.3	94.2 ± 1.1	63.2 ± 1.7	93.5 ± 0.9
	Hopper (High)	54.1 ± 5.5	63.2 ± 11.0	48.4 ± 7.4	18.9 ± 11.2	79.9 ± 3.5	60.0 ± 6.1	88.3 ± 3.5	64.0 ± 16.5	89.9 ± 1.4
	Walker2D (High)	49.9 ± 15.4	44.4 ± 17.6	div.	27.3 ± 13.4	79.5 ± 18.0	31.1 ± 14.9	104.3 ± 2.3	72.7 ± 4.9	107.0 ± 2.3

C.2. HIV Treatment Return Distributions

Figure 3 illustrates the return distributions for each algorithm from all evaluation episodes aggregated over 10 seeds per algorithm. Note that in the ablation in Figure 3c, training became unstable for a subset of the runs. This resulted in very poor performance for a subset of the runs. Because the plots in Figure 3 are aggregated across all 10 runs for each algorithm, this results in the strongly multi-modal performance observed for the ablation in Figure 3c.



(a) 1R2R$_{\text{CVaR}}$

(b) 1R2R$_{\text{Wang}}$

(c) Ablation

(d) ORAAC

(e) CODAC

(f) RAMBO

(g) CQL

(h) IQL

(i) TD3+BC

(j) SAC (Online)

Figure 3: Return distributions for each algorithm on the HIV Treatment domain. The plots are generated by aggregating all evaluation episodes. There are 500 evaluation episodes per iteration for each of the last 10 iterations of training, across 10 seeds per algorithm, for a total of 50,000 evaluation episodes per plot.

C.3. Results with Modified Parameters

In Table 7 we present additional results for 1R2R using parameter values that have been modified from those used in our main experiments. Due to computational limitations, we focus on 1R2R$_{\text{CVaR}}$ on the Stochastic Walker2D domains.

The results in Table 7 show that by either increasing the number of networks in the model ensemble (1R2R$_{\text{CVaR}}$ (Larger Ensemble)), or increasing the number of samples of successor states drawn from the model (1R2R$_{\text{CVaR}}$ (More Samples)), similar or slightly worse performance is obtained compared to the base version of our algorithm. This indicates that it is not possible to improve the performance of 1R2R by naively increasing the computation available.

In the base configuration of our algorithm we set the ratio of real data sampled during policy training to $f = 0.5$, as described in Appendix B.2. The results in Table 7 show that the performance degrades significantly when training the policy and value function entirely from synthetic samples generated by the model (1R2R$_{\text{CVaR}}$ (No Real Data)). This indicates that sampling training data from the real dataset in addition to synthetic data is crucial to obtaining strong performance. This is probably because sampling data from the real dataset helps to mitigate errors due to inaccuracies in the model. As discussed in Section 6, the model may be somewhat inaccurate for these domains due to the restriction of using Gaussian transition functions.

Table 7: Additional results using different parameter values for 1R2R$_{\text{CVaR}}$. The table includes results for both the Expected Value and the CVaR$_{0.1}$ objective, as indicated by the Objective column. The column 1R2R$_{\text{CVaR}}$ (Base) shows the results for 1R2R$_{\text{CVaR}}$ using the base algorithm configuration described in Appendix B. In the remaining columns, the algorithm is modified as follows. 1R2R$_{\text{CVaR}}$ (Larger Ensemble) uses an ensemble of 20 total neural networks with the best 15 selected. 1R2R$_{\text{CVaR}}$ (No Real Data) does not sample any real data when performing updates (i.e. the ratio of real data is $f = 0$). 1R2R$_{\text{CVaR}}$ (More Samples) samples 50 successor states when generating the set of successor states in Line 8 of Algorithm 1 (i.e. $m = 50$).

		Objective	1R2R$_{\text{CVaR}}$ (Base)	1R2R$_{\text{CVaR}}$ (Larger Ensemble)	1R2R$_{\text{CVaR}}$ (No Real Data)	1R2R$_{\text{CVaR}}$ (More Samples)
Medium	Walker2D (Mod)	CVaR$_{0.1}$	29.9 ± 8.7	27.5 ± 7.5	15.0 ± 7.7	26.6 ± 5.1
		Expected Value	67.3 ± 7.8	60.7 ± 9.9	34.0 ± 13.3	58.2 ± 6.9
	Walker2D (High)	CVaR$_{0.1}$	19.4 ± 12.6	4.1 ± 4.7	0.6 ± 0.7	9.5 ± 1.0
		Expected Value	59.5 ± 11.8	38.7 ± 24.4	28.0 ± 4.9	51.4 ± 9.7
Medium Replay	Walker2D (Mod)	CVaR$_{0.1}$	30.1 ± 7.2	27.9 ± 3.5	10.8 ± 14.4	26.8 ± 6.0
		Expected Value	53.9 ± 9.4	53.3 ± 7.2	30.9 ± 19.7	51.2 ± 8.4
	Walker2D (High)	CVaR$_{0.1}$	46.0 ± 10.1	39.1 ± 6.9	1.8 ± 1.7	39.6 ± 6.5
		Expected Value	84.1 ± 5.3	81.8 ± 4.8	26.4 ± 10.8	79.7 ± 7.0
Medium Expert	Walker2D (Mod)	CVaR$_{0.1}$	34.8 ± 6.5	31.7 ± 6.5	7.7 ± 5.6	42.9 ± 16.7
		Expected Value	72.2 ± 7.3	73.5 ± 7.2	22.9 ± 9.6	79.3 ± 11.0
	Walker2D (High)	CVaR$_{0.1}$	10.8 ± 4.3	11.5 ± 1.7	7.3 ± 5.4	12.3 ± 0.6
		Expected Value	49.9 ± 15.4	52.7 ± 9.9	21.3 ± 11.2	50.4 ± 10.5

D. Experiment Details

D.1. Computation Requirements

Each run of 1R2R requires 16-24 hours of computation time on a system with an Intel Core i7-8700 CPU @ 3.20GHz CPU and an NVIDIA GeForce GTX 1070 graphics card.

D.2. Domains

The D4RL MuJoCo domains are from the D4RL offline RL benchmark (Fu et al., 2020). In this work, we introduce the following domains.

Currency Exchange This domain is an adaptation of the Optimal Liquidation Problem (Almgren & Chriss, 2001; Bao & Liu, 2019) to offline RL. We assume that the agent initially holds 100 units of currency A, and wishes to convert all of this to currency B before a deadline T, subject to an exchange rate which changes stochastically. At each time step, the agent may choose how much (if any) of currency A to convert to currency B at the current exchange rate.

An aggressive strategy is to delay converting the currency, in the hope that random fluctuations will lead to a more favourable exchange rate before the deadline T. However, this may lead to a poor outcome if the exchange rate decreases and fails to recover before the deadline. More risk-averse strategies are to either: a) gradually convert the currency over many steps,

Table 8: Expected return received by a randomly initialised policy and an expert policy trained online using SAC. The expert SAC policy is trained online for 2e6 updates for the Stochastic MuJoCo domains, and 5e6 updates for HIVTreatment. In Tables 1, 2, and 3 the values reported are normalised between 0 and 100 using the procedure from Fu et al. (2020) and the values in this table.

	Random Policy	**Expert Policy**
Hopper (Moderate-Noise)	20.1	3049.4
Walker2D (Moderate-Noise)	-1.9	4479.2
Hopper (High-Noise)	11.2	2035.1
Walker2D (High-Noise)	-5.3	4101.3
HIVTreatment	-48.7	5557.9

or b) immediately convert all of the currency even if the current exchange rate is mediocre, to avoid the risk of having to convert the money at even worse exchange rate in the future, to meet the deadline T.

The state space is 3-dimensional, consisting of: the timestep, $t \in \{0, 1, 2, \dots, T\}$; the amount of currency A remaining, $m \in [0, 100]$; and the current exchange rate, $p \in [0, \infty)$. We assume that the exchange rate evolves according to an Ornstein-Uhlenbeck process, which is commonly used for financial modelling (Maller et al., 2009):

$$dp_t = \theta(\mu - p_t)dt + \sigma dW_t, \tag{12}$$

where W_t denotes the Wiener process. In our implementation we set: $\mu = 1.5$, $\sigma = 0.2$, $\theta = 0.05$. The exchange rate at the initial time step is $p_0 \sim \mathcal{N}(1, 0.05^2)$. The action space is single dimensional, $a \in [-1, 1]$. We assume that a positive action corresponds to the proportion of m to convert to currency B at the current step, while a negative action corresponds to converting none of the currency. The reward is equal to the amount of currency B received at each step.

To generate the dataset, we generate data from a random policy that at each step chooses a uniform random proportion of currency to convert with probability 0.2, and converts none of the currency with probability 0.8.

HIV Treatment The environment is based on the implementation in Geramifard et al. (2015) of the physical model described by Ernst et al. (2006). The patient state is a 6-dimensional continuous vector representing the concentrations of different types of cells and viruses in the patients blood. There are two actions, corresponding to giving the patient Reverse Transcriptase Inhibitors (RTI) or Protease Inhibitors (PI) (or both). Unlike Ernst et al. (2006) which considered discrete actions, we modify the domain to allow for continuous actions representing variable quantities of each drug. Following previous work (Keramati et al., 2020), we assume that the treatment at each time step is applied for 20 days, and that there are 50 time steps. Therefore, each episode corresponds to a total period of 1000 days. Also following Keramati et al. (2020), we introduce stochasticity into the domain by adding noise to the efficacy of each drug (ϵ_1 and ϵ_2 in Ernst et al. (2006)). The noise added to the efficacy is normally distributed, with a standard deviation of 15% of the standard value.

For this domain, the dataset is constructed in the same manner as the *Medium-Replay* datasets from D4RL (Fu et al., 2020). Specifically, a policy is trained with SAC (Haarnoja et al., 2018) until it reaches $\approx 40\%$ of the performance of an expert policy. The replay buffer of data collected until reaching this point is used as the offline dataset.

The reward function is that used by Geramifard et al. (2015), except that we scale the reward by dividing it by 1e6. A low concentration of HIV viruses is rewarded, and a cost is associated with the magnitude of treatment applied, representing negative side effects. The average reward received by a random policy and an expert policy trained online using SAC (Haarnoja et al., 2018) can be found in Table 8. In the paper, values are normalised between 0 and 100 using the method proposed by Fu et al. (2020) and the values in Table 8. Thus, a score of 100 represents the average reward of an expert policy, and 0 represents the average reward of a random policy.

We assume that highly risk-averse decision making is desirable in this domain, to avoid some patients experiencing excessive hardship. Therefore, we assume that the objective is to optimise the static $\text{CVaR}_{0.02}$ (i.e. the average total reward in the worst 2% of runs).

Stochastic MuJoCo These benchmarks are obtained by applying random perturbation forces to the original *Hopper* and *Walker2D* MuJoCo domains. We chose to focus on these two domains as there is a risk of the robot falling over before the end of the episode. The perturbation forces represent disturbances due to strong wind or other sources of uncertainty. The force is applied in the x-direction at each step, according to a random walk on the interval $[-f_{\text{MAX}}, f_{\text{MAX}}]$. At each step, the change in force relative to the previous step is sampled uniformly from $\Delta f \sim Unif(-0.1 \cdot f_{\text{MAX}}, 0.1 \cdot f_{\text{MAX}})$.

This is motivated by the fact that the strength of wind is often modelled as a random walk (Popko et al., 2016).

We test using two levels of noise applied. For the *Moderate-Noise* datasets, the level of noise results in a moderate degradation in the performance of an online SAC agent, compared to the original deterministic domain. For the *High-Noise* datasets, the level of noise results in a significant degradation in the performance of an online SAC agent. The value of f_{MAX} is set to the following in each of the domains:

- *Hopper Moderate-Noise*: $f_{\text{MAX}} = 2.5$ Newtons.
- *Hopper High-Noise*: $f_{\text{MAX}} = 5$ Newtons.
- *Walker2D Moderate-Noise*: $f_{\text{MAX}} = 7$ Newtons.
- *Walker2D High-Noise*: $f_{\text{MAX}} = 12$ Newtons.

We construct three datasets: *Medium*, *Medium-Replay*, and *Medium-Expert*. These are generated in the same manner as the D4RL datasets (Fu et al., 2020). The expert policy is a policy trained online using SAC until convergence. The medium policy is trained until it obtains $\approx 40\%$ of the performance of the expert policy. The *Medium-Replay* dataset consists of the replay buffer collected while training the medium policy. The *Medium* dataset contains transitions collected from rolling out the medium policy. The *Medium-Expert* dataset contains a mixture of transitions generated by rolling out both the medium and the expert policy.

Following previous works (Urpí et al., 2021; Ma et al., 2021), we assume that the objective is to optimise the static $\text{CVaR}_{0.1}$ (i.e. the average total reward in the worst 10% of runs).

D.3. Evaluation Procedure

For D4RL MuJoCo, we evaluate the policy for 10 episodes at each of the last 10 iterations of training, for each of 5 seeds. The average normalised total reward received throughout this evaluation is reported in the D4RL results table (Table 1).

For Stochastic MuJoCo, we evaluate the policy for 20 episodes at each of the last 10 iterations of training. $\text{CVaR}_{0.1}$ is computed by averaging the total reward in the worst two episodes at each iteration. The value of $\text{CVaR}_{0.1}$ is then averaged across the last 10 iterations, and across each of the 5 seeds. These are the results reported for $\text{CVaR}_{0.1}$ in Table 3.

For HIVTreatment and CurrencyExchange, we evaluate the policy for 500 episodes at each of the last 10 iterations of training, for each of 10 seeds. For HIVTreatment, $\text{CVaR}_{0.02}$ is computed by averaging the total reward in the worst 10 episodes at each evaluation. The value of $\text{CVaR}_{0.02}$ is then averaged across the last 10 iterations, and across each of the 10 seeds. All of the 50,000 evaluation episodes are aggregated together to generate the return distributions in Figures 2 and 3.

D.4. Baselines

We compare 1R2R against offline RL algorithms designed for risk-sensitive optimisation (O-RAAC (Urpí et al., 2021) and CODAC (Ma et al., 2021)) in addition to performant risk-neutral model-based (RAMBO (Rigter et al., 2022b)) and model-free (IQL (Kostrikov et al., 2022), TD3+BC (Fujimoto & Gu, 2021), and CQL (Kumar et al., 2020)) algorithms.

For the baselines, we use the following implementations:

- O-RAAC: github.com/nuria95/O-RAAC
- CODAC: github.com/JasonMa2016/CODAC
- RAMBO: github.com/marc-rigter/rambo
- IQL: github.com/ikostrikov/implicit_q_learning
- TD3+BC: github.com/sfujim/TD3_BC
- CQL: github.com/takuseno/d3rlpy

The results for D4RL MuJoCo-v2 for RAMBO, IQL, and TD3+BC are taken from the original papers. Because the original CQL paper used the -v0 domains, we report the results for CQL on the -v2 domains from Wang et al. (2021). The results for CODAC in the original paper use D4RL MuJoCo-v0, and O-RAAC does not report results on the D4RL benchmarks. Therefore, we obtain the results for D4RL MuJoCo-v2 for CODAC and O-RAAC by running the algorithms ourselves. The results for all baseline algorithms for the other domains are obtained by running the algorithms ourselves.

D.5. Baseline Hyperparameter Tuning

We tune each of the algorithms online to achieve the best performance for the desired objective in each dataset. The desired optimisation objectives in each domain are: expected value in D4RL MuJoCo; static $\text{CVaR}_{0.02}$ in HIVTreatment; and static $\text{CVaR}_{0.1}$ in Stochastic MuJoCo . O-RAAC and CODAC can be configured to optimise a static risk measure, or expected value. Therefore, we set O-RAAC and CODAC to optimise the appropriate objective for each dataset.

For each of the baseline algorithms, we tune the following parameters based on the best online performance (averaged over 5 seeds for the MuJoCo domains, and 10 seeds for HIVTreatment) for the target objective. Where a hyperparameter value is not specified, it was set to the default value in the corresponding original paper.

- CODAC: The risk measure to optimise is set to the desired objective (Expected value/$\text{CVaR}_{0.02}$/$\text{CVaR}_{0.1}$). For Stochastic MuJoCo and HIVTreatment, we tune the parameters in the same manner is the original paper (Ma et al., 2021), choosing $(\omega, \zeta) \in \{0.1, 1, 10\} \times \{-1, 10\}$. For Stochastic MuJoCo and HIVTreatment we use the default critic learning rate of $\eta_{\text{critic}} = 3e-5$. To generate the results for CODAC on D4RL MuJoCo-v2, we use the same parameters as reported in the paper for D4RL MuJoCo-v0.
- O-RAAC: The risk measure to optimise is set to the desired objective (Expected value/$\text{CVaR}_{0.02}$/$\text{CVaR}_{0.1}$). Following the original paper (Urpí et al., 2021) we tune the imitation learning weighting: $\lambda \in \{0.25, 0.5\}$.
- RAMBO: We tuned the rollout length and adversarial weighting from: : $(k, \lambda) \in \{(1, 3e-4), (2, 3e-4), (5, 3e-4), (5, 0)\}$. This is the same procedure as in the original paper (Rigter et al., 2022b), except that we additionally included a rollout length of 1 as we found this was necessary for RAMBO to train stably on some of the stochastic domains.
- IQL: For Stochastic MuJoCo, we used the same parameters as for D4RL MuJoCo in the original paper (Kostrikov et al., 2022). For HIVTreatment, we chose the best performance between the parameters for D4RL MuJoCo and D4RL AntMaze from the original paper.
- CQL: Following the original paper (Kumar et al., 2020), we tuned the Lagrange parameter, $\tau \in \{5, 10\}$. All other parameters are set to their default values.
- TD3+BC: Following the original paper, we set α to 2.5 (Fujimoto & Gu, 2021).