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 computation for models with dependent uncertainties.

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

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

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, $C : S \times A \times S \rightarrow \mathbb{R}$ is the cost function, and $T : S \times A \times S \rightarrow [0, 1]$ 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*

$$reg(s_0, \pi) = V(s_0, \pi) - V(s_0, \pi^*), \tag{1}$$

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

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

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*

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

$\bar{C}(s, a) - \bar{C}^*(s)$ 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.* $\mathcal{T} = \{T_1, T_2, \dots, T_{|\xi|}\}$ *denotes a finite set of possible transition functions and* $\mathcal{C} = \{C_1, C_2, \dots, C_{|\xi|}\}$ *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*

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

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):

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

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 $\sigma^1 : S \times A \times \xi \rightarrow \{0, 1\}$ which responds to the action of the agent by applying the worst-case parameters at each step:

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

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*

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

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

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

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

Taking $h \rightarrow \infty$ 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 $\sigma^1 : S \times A \times \xi \rightarrow \{0, 1\}$ 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*

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

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:*

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

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

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).

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

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

The results for minimax VI apply (Iyengar 2005; Nilim and El Ghaoui 2005), and therefore the optimal policy is stationary and deterministically chooses options, $\pi^n : S \times O \to \{0, 1\}$. 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 $\kappa \to 0^+$, 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

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

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,*

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

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

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.

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.

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

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

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

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

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

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

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

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

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

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 $\Pi^*_\xi = \{\pi^*_1, \pi^*_2, \dots, \pi^*_{|\xi|}\}$. 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, $h \in \{0, \dots, 19\}$, and the day $d \in \{0, \dots, 6\}$. 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 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

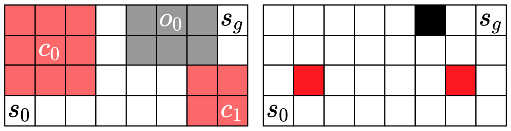


Figure 1: Illustration of an example UMDP in disaster rescue. Left: shaded regions indicate possible swamp and obstacle locations. Right: possible UMDP sample.

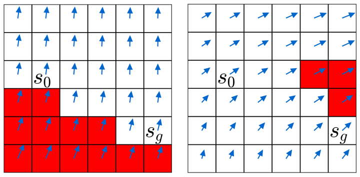
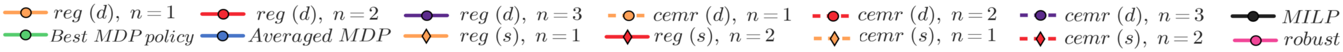


Figure 2: Illustration of two UMDP samples in glider domain. Arrows indicate ocean currents. Red squares indicate states with additional cost.



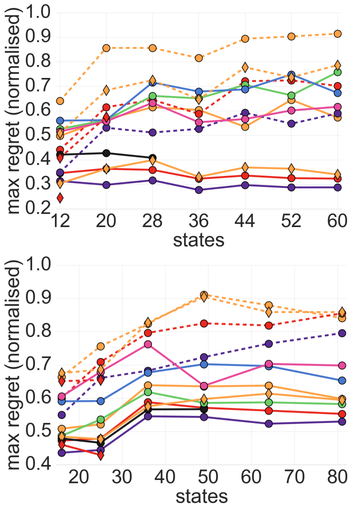


Figure 3: Mean normalised max regret. Top: disaster rescue, bottom: glider.

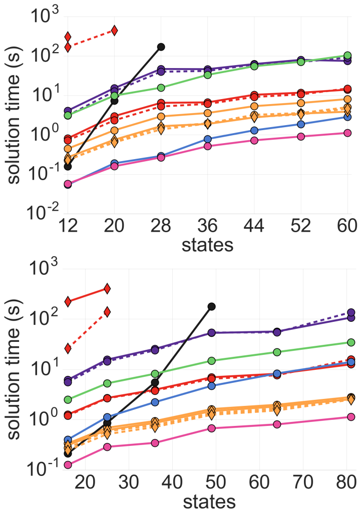


Figure 4: Mean solution times. Top: disaster rescue, bottom: glider.

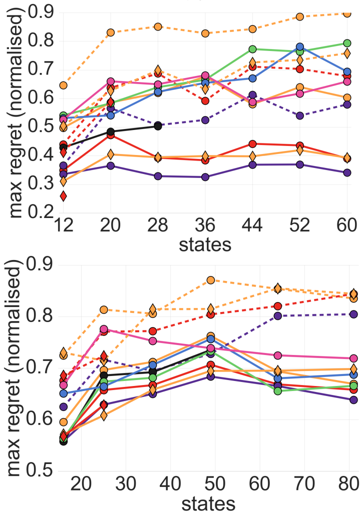


Figure 5: Mean normalised max regret on test set. Top: disaster rescue, bottom: glider.

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.

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.

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.

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* .

Appendices

Proof of Proposition 1

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

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

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

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

$$reg(s_0,\pi) = \sum_{a\in A} \pi(s_0,a)\cdot \left[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot V(s',\pi^*) - V(s_0,\pi^*) + \sum_{s'\in S} T(s_0,a,s')\cdot reg(s',\pi)\right]. \tag{24}$$

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 $reg(s',\pi)$

$$\begin{aligned}reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A}\pi(s_0,a)\cdot\Bigg[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot V(s',\pi^*) +\\ \sum_{s'\in S} T(s_0,a,s')\cdot\Bigg[\sum_{a\in A}\pi(s',a)\cdot\Bigg[\bar{C}(s',a) + \sum_{s''\in S} T(s',a,s'')\cdot V(s'',\pi^*) - V(s',\pi^*) + \sum_{s''\in S} T(s',a,s'')\cdot reg(s'',\pi)\Bigg]\Bigg]\Bigg].\end{aligned} \tag{25}$$

Again, we can move $V(s',\pi^*)$ outside of the inner sum as it does not depend on a. Thus, Equation 25 can be rewritten as

$$\begin{aligned}reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A}\pi(s_0,a)\cdot\Bigg[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot V(s',\pi^*) +\\ \sum_{s'\in S} T(s_0,a,s')\cdot\Bigg[-V(s',\pi^*) + \sum_{a\in A}\pi(s',a)\cdot\Bigg[\bar{C}(s',a) + \sum_{s''\in S} T(s',a,s'')\cdot V(s'',\pi^*) + \sum_{s''\in S} T(s',a,s'')\cdot reg(s'',\pi)\Bigg]\Bigg]\Bigg].\end{aligned} \tag{26}$$

Cancelling terms, we have

$$\begin{aligned}reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A}\pi(s_0,a)\cdot\Bigg[\bar{C}(s_0,a) +\\ \sum_{s'\in S} T(s_0,a,s')\cdot\Bigg[\sum_{a\in A}\pi(s',a)\cdot\Bigg[\bar{C}(s',a) + \sum_{s''\in S} T(s',a,s'')\cdot V(s'',\pi^*) + \sum_{s''\in S} T(s',a,s'')\cdot reg(s'',\pi)\Bigg]\Bigg]\Bigg].\end{aligned} \tag{27}$$

After repeating the above process of unrolling the expression and cancelling terms for h steps we arrive at the following expression

$$\begin{aligned}reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A}\pi(s_0,a)\cdot\Bigg[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot\Bigg[\sum_{a\in A}\pi(s',a)\cdot\Bigg[\bar{C}(s',a) + \dots\\ \sum_{s^{h\text{-}1}\in S} T(s^{h\text{-}2},a,s^{h\text{-}1})\cdot\Bigg[\sum_{a\in A}\pi(s^{h\text{-}1},a)\cdot\Bigg[\bar{C}(s^{h\text{-}1},a) + \sum_{s^h\in S} T(s^{h\text{-}1},a,s^h)\cdot V(s^h,\pi^*) + \sum_{s^h\in S} T(s^{h\text{-}1},a,s^h)\cdot reg(s^h,\pi)\Bigg]\Bigg]\dots\Bigg]\Bigg]\Bigg].\end{aligned} \tag{28}$$

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

$$\begin{aligned} reg(s_0,\pi) = -V(s_0,\pi^*) + \sum_{a\in A}\pi(s_0,a)\cdot\Big[\bar{C}(s_0,a) + \sum_{s'\in S} T(s_0,a,s')\cdot\Big[\sum_{a\in A}\pi(s',a)\cdot\Big[\bar{C}(s',a)+\dots \\ \sum_{s^{h-1}\in S} T(s^{h-2},a,s^{h-1})\cdot\Big[\sum_{a\in A}\pi(s^{h\text{-}1},a)\cdot\bar{C}(s^{h\text{-}1},a)\Big]\dots\Big]\Big]\Big]. \end{aligned} \tag{29}$$

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. $\square$

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 $\text{argmin}_{\pi\in\Pi} \max_{\xi_q\in\xi} reg_q(s_0, \pi) = \text{argmin}_{\pi\in\Pi} \max_{\sigma^1} reg(s_0, \pi)$. $\square$

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:*

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

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

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

$$\max_{\sigma^1} reg(s_0,\pi) \geq \max_{\xi_q\in\xi} reg_q(s_0,\pi) \implies \max_{\sigma^1} reg(s_0,\pi) - \max_{\xi_q\in\xi} reg_q(\pi) \geq 0. \tag{30}$$

To prove the upper bound, we begin by denoting the most adversarial uncertainty sample by $\xi_\top = (C_\top, T_\top) = \text{argmax}_{\xi_q\in\xi} reg_q(s_0, \pi)$. 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

$$f(s) = \max_{\sigma^1} reg(s,\pi) - reg_\top(s,\pi). \tag{31}$$

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

$$
\begin{aligned}
f(s) &= \max_{\sigma^1} reg(s,\pi) - reg_\top(s,\pi) \\
&= \max_{\xi_q \in \xi} \Bigg[\sum_{a \in A} \pi(s,a) \cdot \Bigg[\bar{C}_q(s,a) + \sum_{s' \in S} T_q(s,a,s') \cdot V_q(s',\pi^*) - V_q(s,\pi^*) + \\
&\qquad \sum_{s' \in S} T_q(s,a,s') \cdot \max_{\sigma^1} reg(s',\pi) \Bigg]\Bigg] - \Bigg[\sum_{a \in A} \pi(s,a) \cdot \Bigg[\bar{C}_\top(s,a) + \sum_{s' \in S} T_\top(s,a,s') \cdot V_\top(s',\pi^*) \\
&\qquad - V_\top(s,\pi^*) + \sum_{s' \in S} T_\top(s,a,s') \cdot reg_\top(s',\pi) \Bigg]\Bigg]. \qquad (32)
\end{aligned}
$$

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

$$
\begin{aligned}
\Delta T_q(s,a,s') &= T_q(s,a,s') - T_\top(s,a,s'), \\
\Delta V_q(s,\pi^*) &= V_q(s,\pi^*) - V_\top(s,\pi^*).
\end{aligned}
$$

We also make a substitution for $\max_{\sigma^1} reg(s',\pi)$ using the definition of $f(s')$ from Equation 31. This results in the following expression

$$
\begin{aligned}
f(s) = \max_{\xi_q \in \xi} \Bigg[&\sum_{a \in A} \pi(s,a) \cdot \Bigg[\bar{C}_q(s,a) + \sum_{s' \in S} (T_\top(s,a,s') + \Delta T_q(s,a,s')) \cdot (V_\top(s',\pi^*) + \Delta V_q(s',\pi^*)) - \\
&V_\top(s,\pi^*) - \Delta V_q(s,\pi^*) + \sum_{s' \in S} (T_\top(s,a,s') + \Delta T_q(s,a,s')) \cdot (reg_\top(s',\pi) + f(s')) \Bigg]\Bigg] - \\
&\Bigg[\sum_{a \in A} \pi(s,a) \cdot \Bigg[\bar{C}_\top(s,a) + \sum_{s' \in S} T_\top(s,a,s') \cdot V_\top(s',\pi^*) - V_\top(s,\pi^*) + \sum_{s' \in S} T_\top(s,a,s') \cdot reg_\top(s',\pi) \Bigg]\Bigg]. \qquad (33)
\end{aligned}
$$

Cancelling terms and expanding, we have

$$
\begin{aligned}
f(s) = \max_{\xi_q \in \xi} \Bigg[&\sum_{a \in A} \pi(s,a) \cdot \Bigg[\bar{C}_q(s,a) + \sum_{s' \in S} T_q(s,a,s') \cdot \Delta V_q(s',\pi^*) + \sum_{s' \in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) - \Delta V_q(s,\pi^*) + \\
&\sum_{s' \in S} T_q(s,a,s') \cdot f(s') + \sum_{s' \in S} \Delta T_q(s,a,s') \cdot reg_\top(s',\pi) \Bigg]\Bigg] - \sum_{a \in A} \pi(s,a) \cdot \bar{C}_\top(s,a). \qquad (34)
\end{aligned}
$$

At this point, we start to upper bound the terms in the previous equation using the constants defined in Proposition 3:

$$
\begin{aligned}
f(s) \leq \delta_C + 2\delta_{V^*} + \max_{\xi_q \in \xi} \Bigg[&\sum_{a \in A} \pi(s,a) \Bigg[\sum_{s' \in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) + \\
&\sum_{s' \in S} T_q(s,a,s') \cdot f(s') + \sum_{s' \in S} \Delta T_q(s,a,s') \cdot reg_\top(s',\pi) \Bigg]\Bigg]. \qquad (35)
\end{aligned}
$$

We observe that

$$
\sum_{s' \in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) \leq \delta_T \max_{s' \in S} V_\top(s',\pi^*) - \delta_T \min_{s' \in S} V_\top(s',\pi^*). \qquad (36)
$$

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

$$
0 \leq V_\top(s',\pi^*) \leq V_\top(s',\pi) \leq C_{max} H. \qquad (37)
$$

Therefore

$$
\sum_{s' \in S} \Delta T_q(s,a,s') \cdot V_\top(s',\pi^*) \leq \delta_T C_{max} H. \qquad (38)
$$

By definition, $reg_{\top}(s, \pi) = V_{\top}(s, \pi) - V_{\top}(s, \pi^*)$, and can therefore similarly be bounded

$$0 \leq reg_{\top}(s', \pi) \leq C_{max}H, \tag{39}$$

$$\sum_{s' \in S} \Delta T_q(s, a, s') \cdot reg_{\top}(s', \pi^*) \leq \delta_T C_{max} H. \tag{40}$$

Combining with Equation 35 gives

$$f(s) \leq \delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H + \max_{\xi_q \in \xi} \bigg[\sum_{a \in A} \pi(s, a) \cdot \Big[\sum_{s' \in S} T_q(s, a, s') \cdot f(s') \Big] \bigg]. \tag{41}$$

Thus, we can write for the initial state

$$\begin{aligned} f(s_0) \leq{}& \delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H + \\ & \max_{\xi_q \in \xi} \bigg[\sum_{a \in A} \pi(s_0, a) \cdot \Big[\sum_{s' \in S \setminus G} T_q(s_0, a, s') \cdot f(s') + \sum_{s'_g \in G} T_q(s_0, a, s'_g) \cdot f(s'_g) \Big] \bigg] \\ \leq{}& \Pr^{\pi, \sigma^1}(\tau^G_{s_0} = 1) \cdot (\delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H) + \\ & \Pr^{\pi, \sigma^1}(\tau^G_{s_0} \geq 2) \cdot \Big[\delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H + \max_{s' \in S} f(s') \Big], \end{aligned} \tag{42}$$

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 41 to substitute for $\max_{s' \in S} f(s')$ and repeatedly applying the same reasoning we have

$$f(s_0) \leq (\delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H) \sum_{h=1}^{\infty} \Pr^{\pi, \sigma^1}(\tau^G_{s_0} = h) \cdot h \tag{43}$$

The sum on the right hand side is simply the expected number of steps to reach the goal and therefore we have the result

$$f(s_0) \leq (\delta_C + 2\delta_{V^*} + 2\delta_T C_{max} H) H. \quad \square \tag{44}$$

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.

$$\pi_{robust}(n\text{-}MDP) = \operatorname*{argmin}_{\pi \in \Pi} \max_{\sigma^n} V(s_0, \pi), \text{ where} \tag{45}$$

$$V(s, \pi^n) = \sum_{o \in O} \pi^n(s, o) \cdot \Big[C^o(s, o) + \sum_{s' \in S} T(s, o, s') \cdot V(s', \pi^n) \Big]. \tag{46}$$

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

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

We unroll the definition for n steps, and cancel terms in the same manner as in the proof of Proposition 1.

$$reg(s,\pi) = \sum_{a\in A} \pi(s,a) \cdot \Bigg[\bar{C}(s,a) - V(s,\pi^*) + \sum_{s'\in S} T(s,a,s') \cdot \Bigg[\sum_{a\in A} \pi(s',a) \cdot \Bigg[\bar{C}(s',a) + \dots$$

$$\sum_{s^{n\text{-}1}\in S} T(s^{n\text{-}2},a,s^{n\text{-}1}) \cdot \Bigg[\sum_{a\in A} \pi(s^{n\text{-}1},a) \cdot \Bigg[\bar{C}(s^{n\text{-}1},a) + \sum_{s^n\in S} T(s^{n\text{-}1},a,s^n) \cdot V(s^n,\pi^*) + \sum_{s^n\in S} T(s^{n\text{-}1},a,s^n) \cdot reg(s^n,\pi)\Bigg]\Bigg] \dots \Bigg]\Bigg]\Bigg]. \tag{47}$$

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.

$$reg(s,\pi^n) = \sum_{o\in O} \pi^n(s,o) \cdot \sum_{a\in A} \pi^o(s,a) \cdot \Bigg[\bar{C}(s,a) - V(s,\pi^*) + \sum_{s'\in S} T(s,a,s') \cdot \Bigg[\sum_{a\in A} \pi^o(s',a) \cdot \Bigg[\bar{C}(s',a) + \dots$$

$$\sum_{s^{n\text{-}1}\in S} T(s^{n\text{-}2},a,s^{n\text{-}1}) \cdot \Bigg[\sum_{a\in A} \pi^o(s^{n\text{-}1},a) \cdot \Bigg[\bar{C}(s^{n\text{-}1},a) + \sum_{s^n\in S} T(s^{n\text{-}1},a,s^n) \cdot V(s^n,\pi^*) + \sum_{s^n\in S} T(s^{n\text{-}1},a,s^n) \cdot reg(s^n,\pi^n)\Bigg]\Bigg] \dots \Bigg]\Bigg]\Bigg]. \tag{48}$$

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

$$reg(s,\pi^n) = \sum_{o\in O} \pi^n(s,o) \cdot \Bigg[V^n(s,\pi^o) + \sum_{s'\in S} T(s,o,s') \cdot V(s',\pi^*) - V(s,\pi^*) + \sum_{s'\in S} T(s,o,s') \cdot reg(s',\pi^n)\Bigg]. \tag{49}$$

Substituting the cost function, $C^o(s,o)$ given by Equation 12 we can now write Problem 3 as

$$\pi^n_{reg} = \underset{\pi\in\Pi}{\text{argmin}} \max_{\sigma^n} reg(s_0,\pi^n), \text{where}$$

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

We observe that this statement of Problem 3 is identical to the formulation of the robust policy for the n-UMDP in Equations 45-46. 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

$$\max_{\sigma^n} reg(s_0,\pi) \geq \max_{\xi_q\in\xi} reg_q(s_0,\pi) \implies \max_{\sigma^n} reg(s_0,\pi) - \max_{\xi_q\in\xi} reg_q(\pi) \geq 0. \tag{50}$$

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

$$\min_{\pi^n} \max_{\sigma^n} reg(s,\pi^n) = \min_{\pi^n} \max_{\xi_1\in\xi} \Bigg[\sum_{o\in O} \pi^n(s,o) \cdot \Bigg[C^o_1(s,o) + \sum_{s^n\in S} T^o_1(s,o,s^n) \cdot \Bigg[\dots$$

$$+ \sum_{s^{(k-1)n}\in S} T^o_{k-1}(s^{(k-2)n},o,s^{(k-1)n}) \cdot \Bigg[\max_{\xi_k\in\xi} \Bigg[\sum_{o\in O} \pi^n(s^{(k-1)n},o) \cdot \Bigg[C^o_k(s^{(k-1)n},o) +$$

$$\sum_{s^{kn}\in S} T^o_k(s^{(k-1)n},o,s^{kn}) \cdot \max_{\sigma^n} reg(s^{kn},\pi^n)\Bigg]\Bigg]\Bigg] \dots \Bigg]\Bigg]\Bigg], \tag{51}$$

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

$$\begin{aligned}\min_{\pi^n}\max_{\sigma^{kn}} reg(s,\pi^n) = \min_{\pi^n}\max_{\xi_q\in\xi}\Bigg[\sum_{o\in O}\pi^n(s,o)\cdot\Bigg[C_q^o(s,o)+\sum_{s^n\in S}T_q^o(s,o,s^n)\cdot\Bigg[\dots \\ +\sum_{s^{(k-1)n}\in S}T_q^o(s^{(k-2)n},o,s^{(k-1)n})\cdot\Bigg[\sum_{o\in O}\pi^n(s^{(k-1)n},o)\cdot\Bigg[C_q^o(s^{(k-1)n},o)+ \\ \sum_{s^{kn}\in S}T_q^o(s^{(k-1)n},o,s^{kn})\cdot\max_{\sigma^{kn}} reg(s^{kn},\pi^n)\Bigg]\Bigg]\dots\Bigg]\Bigg]\Bigg].\end{aligned} \tag{52}$$

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

$$\min_{\pi^n}\max_{\sigma^n} reg(s,\pi^n) \geq \min_{\pi^n}\max_{\sigma^{kn}} reg(s,\pi^n) \quad \forall\, n,k\in\mathbb{N}. \tag{53}$$

Additionally, we have that

$$\min_{\pi^n}\max_{\sigma^{kn}} reg(s,\pi^n) \geq \min_{\pi^{kn}}\max_{\sigma^{kn}} reg(s,\pi^{kn}) \quad \forall\, n,k\in\mathbb{N}, \tag{54}$$

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 53 and 54 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 55-58

$$c_q(s,t) = \textstyle\sum_a c_q'(s,a,t) \qquad \forall s\in S^q_{\bar{s},t},\xi_q,t\leq n-1 \tag{55}$$

$$c_q'(s,a,t) \leq c_q(s,a,t) \qquad \forall s\in S^q_{\bar{s},t},a,\xi_q,t\leq n-1 \tag{56}$$

$$c_q'(s,a,t) \leq \pi^o(s,a)\cdot M \qquad \forall s\in S^q_{\bar{s},t},a,\xi_q,t\leq n-1 \tag{57}$$

$$c_q'(s,a,t) \geq c_q(s,a,t) - (1-\pi^o(s,a))\cdot M \qquad \forall s\in S^q_{\bar{s},t},a,\xi_q,t\leq n-1 \tag{58}$$

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:

$$c_q(s,t) = \sum_a \left[x_q(s,a,t)^2 - y_q(s,a,t)^2\right] \qquad \forall s\in S^q_{\bar{s},t},\xi_q,t\leq n-1 \tag{59}$$

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

$$x_q(s,a,t) = \sum_i \lambda_q^i(s,a,t) \cdot b_i \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{60}$$

$$x_q(s,a,t)^2 = \sum_i \lambda_q^i(s,a,t) \cdot b_i^2 \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{61}$$

$$\sum_i \lambda_q^i(s,a,t) = 1 \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{62}$$

$$SOS2(\{\lambda_q^i(s,a,t) | 0 \leq i \leq m\}) \qquad \forall s \in S^q_{\bar{s},t}, a, \xi_q, t \leq n-1 \tag{63}$$

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: $\mathbf{x} \rightarrow s$ 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.[1] 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

---

[1] 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

—🟠— *reg* $(d),\ n=1$ —🔴— *reg* $(d),\ n=2$ —🟣— *reg* $(d),\ n=3$ - 🟠 - *cemr* $(d),\ n=1$ - 🔴 - *cemr* $(d),\ n=2$ - 🟣 - *cemr* $(d),\ n=3$ —⚫— *MILP*
—🟢— *Best MDP policy* —🔵— *Averaged MDP* —🔶— *reg* $(s),\ n=1$ —♦— *reg* $(s),\ n=2$ - 🔶 - *cemr* $(s),\ n=1$ - ♦ - *cemr* $(s),\ n=2$ —🩷— *robust*

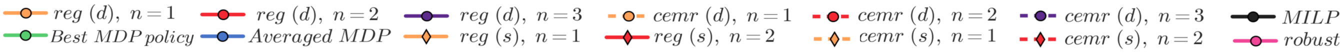
Method	—🟠—	—🔴—	—🟣—	- 🟠 -	- 🔴 -	- 🟣 -	—🟢—	—🔵—	—🩷—	—🔶—	- 🔶 -
max reg	0.576; 0.25	0.340; 0.19	**0.297**; 0.17	0.840; 0.25	0.633; 0.25	0.521; 0.24	0.647; 0.25	0.661; 0.28	0.578; 0.27	0.353; 0.18	0.695; 0.23
—🟠—	-	-	-	< 0.0001	0.017	-	0.004	0.001	0.47	-	< 0.0001
—🔴—	< 0.0001	-	-	< 0.0001	< 0.0001	< 0.0001	< 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

—🟠— $reg\ (d),\ n=1$
—🔴— $reg\ (d),\ n=2$
—🟣— $reg\ (d),\ n=3$
- 🟠 - $cemr\ (d),\ n=1$
- 🔴 - $cemr\ (d),\ n=2$
- 🟣 - $cemr\ (d),\ n=3$
—⚫— $MILP$
—🟢— $Best\ MDP\ policy$
—🔵— $Averaged\ MDP$
—🔶— $reg\ (s),\ n=1$
—♦— $reg\ (s),\ n=2$
- 🔶 - $cemr\ (s),\ n=1$
- ♦ - $cemr\ (s),\ n=2$
—🩷— $robust$

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.