Journal of Artificial Intelligence Research 59 (2017) 229-264 Submitted 06/16; published 06/17

Sampling Based Approaches for Minimizing Regret in Uncertain Markov Decision Processes (MDPs)

Asrar Ahmed AMOHAMMED@LONDON.EDU
London Business School

Pradeep Varakantham PRADEEPV@SMU.EDU.SG
Meghna Lowalekar MEGHNAL.2015@PHDIS.SMU.EDU.SG
School of Information Systems
Singapore Management University

Yossiri Adulyasak YOSSIRI.ADULYASAK@HEC.CA
Department of Logistics and Operations Management
HEC Montréal and GERAD

Patrick Jaillet JAILLET@MIT.EDU
Department of Electrical Engineering and Computer Science
Massachussets Institute of Technology

Abstract

Markov Decision Processes (MDPs) are an effective model to represent decision processes in the presence of transitional uncertainty and reward tradeoffs. However, due to the difficulty in exactly specifying the transition and reward functions in MDPs, researchers have proposed uncertain MDP models and robustness objectives in solving those models. Most approaches for computing robust policies have focused on the computation of *maximin* policies which maximize the value in the worst case amongst all realisations of uncertainty. Given the overly conservative nature of *maximin* policies, recent work has proposed *minimax* regret as an ideal alternative to the *maximin* objective for robust optimization. However, existing algorithms for handling *minimax* regret are restricted to models with uncertainty over rewards only and they are also limited in their scalability. Therefore, we provide a general model of uncertain MDPs that considers uncertainty over both transition and reward functions. Furthermore, we also consider dependence of the uncertainty across different states and decision epochs. We also provide a mixed integer linear program formulation for minimizing regret given a set of samples of the transition and reward functions in the uncertain MDP. In addition, we provide two myopic variants of regret, namely Cumulative Expected Myopic Regret (CEMR) and One Step Regret (OSR) that can be optimized in a scalable manner. Specifically, we provide dynamic programming and policy iteration based algorithms to optimize CEMR and OSR respectively. Finally, to demonstrate the effectiveness of our approaches, we provide comparisons on two benchmark problems from literature. We observe that optimizing the myopic variants of regret, OSR and CEMR are better than directly optimizing the regret.

1. Introduction

For a multitude of reasons, ranging from dynamic environments to conflicting elicitations from experts, from insufficient data to aggregation of states in exponentially large problems, it is difficult to exactly specify reward and transition functions in Markov Decision Processes

(MDPs). Motivated by this difficulty in exact specification, researchers have proposed uncertain MDP models and robustness objectives in solving these models.

A robust solution typically provides guarantees on the worst case performance. Most of the research in computing robust solutions has assumed a *maximin* objective, where one computes a policy that maximizes the value corresponding to the worst case realization (Nilim & Ghaoui, 2005; Iyengar, 2005; Givan, Leach, & Dean, 2000; Bagnell, Ng, & Schneider, 2001; Mastin & Jaillet, 2012). This line of work has developed scalable algorithms by exploiting independence of uncertainties across states and convexity of uncertainty sets. Recently, techniques have been proposed to deal with dependence of uncertainties (Wiesemann, Kuhn, & Rustem, 2013; Mannor, Mebel, & Xu, 2012). This notion of robustness can be viewed as a game against the environment, where given a policy, the environment is choosing an instantiation of transition and reward functions that will minimize the expected value.

Delage and Mannor (2010) and others have highlighted the conservative nature of *maximin* policies. To address this issue, Regan and Boutilier (2009) and Xu and Mannor (2009) have proposed *minimax* regret criterion (Savage, 1954) as an alternative to *maximin* objective for uncertain MDPs. Regret associated with a policy, π and an instantiation, ξ_q is defined as the difference between optimal expected value for an instantiation and expected value of π for that same instantiation. Thus, in *minimax* criterion, the goal is to find a policy that has the least value of maximum regret over all instantiations of uncertainty. This notion of robustness can be treated as a game against the environment, where the environment is choosing an instantiation of uncertainty so as to maximize the regret. In this paper, we also focus on this *minimax* notion of robustness.

While *minimax* regret policies are not conservative, computing optimal *minimax* regret policies is NP-Hard (Xu & Mannor, 2009) and hence is not scalable. Existing algorithms (Regan & Boutilier, 2010; Xu & Mannor, 2009; Regan & Boutilier, 2009) have only focused on computing optimal *minimax* regret solutions to uncertain MDPs, where only the reward function is uncertain. Furthermore, the uncertainties in reward for states and decision epochs are assumed to be independent of each other. In this paper, we provide a general model that not only considers uncertainty over both reward and transition functions, but also considers the dependency in uncertainty across states, decision epochs.

Recent research in planning under uncertainty has demonstrated that sampling-based techniques (Kearns, Mansour, & Ng, 2002; Pineau, Gordon, & Thrun, 2003) are not only efficient but also provide apriori (Chernoff-Hoeffiding bounds) and posteriori (Shapiro, 2003a) quality bounds. Hence, we focus on sampling based approaches for direct and indirect minimisation of maximum regret:

- Approximate the computation of *minimax* regret policy for a given set of samples.
- Optimize alternative but scalable variants of regret, namely Cumulative Expected Myopic Regret (CEMR) and One Step Regret (OSR) for a given set of samples.

Along these two directions, we make the following key methodological contributions:

- An approximate Mixed Integer Linear Programming (MILP) formulation for computing *minimax* regret policies. We also provide guarantees on the quality of the policies obtained using this approximate MILP.

- By exploiting the optimal substructure property of CEMR, we provide a dynamic programming approach to minimize CEMR.
- Relying on a novel combination of value and flow variables, we provide a scalable policy iteration based algorithm that computes locally optimal OSR policies.

We evaluate the performance of our approaches on two benchmark problems from literature, namely stochastic inventory control and disaster rescue. The results show that our policy iteration based algorithm that optimizes OSR outperforms the other approaches consistently with respect to minimax regret. In stochastic inventory control, where we have a scalable approach to optimize CEMR (DP-CEMR), that approach performs better than all other approaches except the algorithm optimizing OSR. Overall, on the two benchmarks considered, optimizing myopic variants of regret is better than direct approximation of regret. We also perform a Sample Average Approximation (SAA) analysis (details provided in section 8.3) to compute posteriori bounds on the solutions obtained using a fixed set of training samples.

In Section 2, we provide background on the MDP model and the Linear Programming (LP) approach for solving MDPs. We then provide preliminary definitions and notations in Section 3 in order to describe the generic uncertain MDP model in Section 4. We describe our approaches to solve the uncertain MDP model with respect to regret, CEMR and OSR in Sections 5, 6 and 7, respectively. Experimental setup is described in Section 8 and the experimental results are shown in Section 9. Sections 10 and 11 describe the related work and conclusions, respectively.

2. Background: Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) (Puterman, 1994) are used to represent decision problems in the face of transitional uncertainty. An MDP is defined using the following tuple:

$$\langle \mathcal{S}, \mathcal{A}, T, R, H, \alpha \rangle$$

$\mathcal{S}$ represents the set of states, $\mathcal{A}$ represents the set of actions or decisions that can be taken in the states, $T : \mathcal{S} \times \mathcal{A} \times H \rightarrow \Delta(\mathcal{S})$ represents the transition function and is a probability distribution over the destination states. $T^t(s,a,s')$ is the probability that the state transitions from s to s' on taking action a at decision epoch t and $\sum_{s'} T^t(s,a,s') = 1, \forall s,a,t$. $R : \mathcal{S} \times \mathcal{A} \times H \rightarrow \mathbb{R}$ represents the reward matrix. $R^t(s,a)$ is the reward obtained by taking action a in state s. α is the starting distribution over states and H is the time horizon.

The goal is to obtain a policy, $\vec{\pi}^0 : \mathcal{S} \times H \rightarrow \Delta(\mathcal{A})$ such that the expected value, $v^0(\alpha)$ is maximized for the time horizon H, given a starting distribution α over states. $\Delta(\mathcal{A})$ denotes a probability distribution over the set of actions $\mathcal{A}$. More concretely, $\pi^t(s,a)$ gives the probability of taking action a in state s at time t and $\sum_a \pi^t(s,a) = 1$. v^0 is characterized using the following expression:

$$v^0(\alpha) = \max_{\pi} \sum_{s \in \mathcal{S}} v^0(s, \pi^0) \cdot \alpha(s), \text{ where}$$

$$v^t(s, \pi^t) = \begin{cases} \sum_a \pi^t(s,a) \cdot R^t(s,a) & t = H-1 \\ \sum_a \pi^t(s,a) \cdot \left(R^t(s,a) + \sum_{s'} T^t(s,a,s') \cdot v^{t+1}(s', \pi)\right) & \text{otherwise} \end{cases} \quad (1)$$

The optimal policy in the case of an MDP is a deterministic one (Puterman, 1994). That is to say, for all time steps t and for all states s there exists one action a, such that $\pi^t(s,a) = 1$.

2.1 Solving MDPs

A number of exact and approximate techniques have been published in the literature to solve MDPs (Bellman, 1957; Howard, 1960; Barto, Bradtke, & Singh, 1995). Since our approaches in this paper are based on linear optimization, we only describe the LP approaches for solving MDPs.

2.1.1 Linear Programming (LP) Approaches

The standard primal formulation to compute the optimal policy involves solving the following linear program.

$$\min_{v^0} \sum_s \alpha(s) \cdot v^0(s) \tag{2}$$

$$\textbf{s.t.} \quad v^t(s,a) = R^t(s,a) \qquad \forall s,a,t = H-1 \tag{3}$$

$$v^t(s,a) = R^t(s,a) + \gamma \sum_{s'} T^t(s,a,s') \cdot v^{t+1}(s') \qquad \forall s,a,t < H-1 \tag{4}$$

$$v^t(s) \geq v^t(s,a) \qquad \forall s,a,t \tag{5}$$

Constraint (4) computes the payoff for each action given future expected payoff. Constraint (5) ensures that the value function at each time step and each state is assigned to the maximum value across all actions in that state. Minimizing the function guarantees the value is bounded from above. Once the value function is computed the policy is obtained as follows:

$$\pi^t(s,a) = \begin{cases} 1 & \text{if} \quad a = \text{argmax}'_a \left\{ v^t(s,a') \right\} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The policy can also be computed using the well-known dual LP optimization (Puterman, 1994). It should be noted that in the dual formulation, the decision variables are state action frequencies which are used in the computation of expected value instead of using value function variables as shown in Equation (1)).

$$\begin{aligned}
&\max_x \sum_{t,s,a} R^t(s,a) \cdot x^t(s,a) \\
&\text{s.t.} \quad \sum_a x^0(s',a) = \alpha(s'), && \forall s' \\
&\qquad \sum_a x^{t+1}(s',a) - \sum_{s,a} x^t(s,a) \cdot T^t(s,a,s') = 0, && \forall\ 0 < t \leq H-1, \forall s' \\
&\qquad x^t(s,a) \geq 0, && \forall s,a,t;
\end{aligned}$$

where $x^t(s,a)$ represents the number of times action a has been chosen in time t in state s. The agent policy for each state and time step can then be obtained by normalizing $\{x^t(s,a)\}$:

$$\pi = \left\{ \pi^t(s,a) : \pi^t(s,a) = \frac{x^t(s,a)}{\sum_{a'} x^t(s,a')}, \forall t, s \right\}.$$

Since x represents the number of times actions are executed in the states, the total value is obtained by a dot product of x and R vectors. The first and second constraints are for flow preservation. That is to say, flow into the state is equal to the flow out of the state. While the optimization problem allows for probabilistic or mixed policies, the solution obtained by maximizing expected value is always deterministic. That is to say,

$$\forall\ s, t; \exists\ a \quad \textbf{s.t.} \quad \pi^t(s,a) = 1$$

A key advantage of the dual approach is that the state action frequencies, which are of relevance to the techniques in this paper are computed directly. We later discuss how both formulations can be used simultaneously in minimizing OSR.

3. Preliminaries

We now formally define the three regret criteria that will be employed in this paper. In the definitions below, we assume an underlying MDP, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, H, \alpha \rangle$, a policy is represented as: $\vec{\pi}^t = \{\pi^t, \pi^{t+1}, \ldots, \pi^{H-1}\}$, the optimal policy is represented as $\vec{\pi}^*$, and the optimal expected value is represented as $v^0(\vec{\pi}^*)$. The maximum reward in any state s at time step t is denoted as $R^{*,t}(s) = \max_a R^t(s,a)$ and minimum reward in any state s at time step t is denoted as $R^{',t}(s)$. Throughout the paper, we use $\alpha(s)$ to denote the starting state distribution in state s and γ to represent the discount factor.

Definition 1 *Regret for any policy $\vec{\pi}^0$ is the difference between the value of policy $\vec{\pi}^0$ and the value of optimal policy. The regret value is denoted by $reg(\vec{\pi}^0)$ and is defined as:*

$$reg(\vec{\pi}^0) = v^0(\vec{\pi}^*) - v^0(\vec{\pi}^0), \textit{\textbf{where}}$$

$$v^0(\vec{\pi}^0) = \sum_s \alpha(s) \cdot v^0(s, \vec{\pi}^0),$$
$$v^t(s, \vec{\pi}^t) = \sum_a \pi^t(s,a) \cdot \Big[R^t(s,a) + \gamma \sum_{s'} T^t(s,a,s') \cdot v^{t+1}(s', \vec{\pi}^{t+1}) \Big]$$

While regret is a well-established robustness concept, calculating policies that minimize maximum regret is known to be computationally challenging when there is uncertainty associated with reward or transition functions (Xu & Mannor, 2009). One of the approaches we employ in this paper is to optimize other variants of regret, with the underlying assumption (validated by our experimental results) that optimizing these variants of regret will yield low regret solutions. We first define the variants of regret, which are myopic and consequently have scalable algorithms to optimize them. The first of these myopic variants is called the Cumulative Expected Myopic Regret (CEMR) which is defined as follows:

Definition 2 *CEMR for policy $\vec{\pi}^0$ is denoted by $cemr(\vec{\pi}^0)$ and is defined as:*

$$cemr(\vec{\pi}^0) = \sum_s \alpha(s) \cdot cemr^0(s, \vec{\pi}^0), \quad \textbf{\textit{where}}$$
$$cemr^t(s, \vec{\pi}^t) = \sum_a \pi^t(s,a) \cdot \left[R^{*,t}(s) - R^t(s,a) + \gamma \sum_{s'} T^t(s,a,s') \cdot cemr^{t+1}(s', \vec{\pi}^{t+1})\right] \quad (7)$$

Regret (as described in Definition 1) for a given policy denotes the total improvement possible by switching to an optimal policy (w.r.t maximizing expected value) from that policy. On the other hand, CEMR for a policy is the cumulation of expected "maximum" possible improvements in each state for that policy. Since we aggregate improvements at individual states (and not consider overall improvement for the entire decision process), CEMR is a myopic variant of regret. Furthermore, it should be noted that the "maximum" improvements considered by CEMR in every state may not be feasible in any optimal policy. Also, the given policy may have significantly different transition dynamics to an optimal policy and hence CEMR for a given policy is not guaranteed to be an upper or lower bound of actual regret.

The following proposition highlights the mathematical dependencies between regret and CEMR. Intuitively, the key connection is through the maximum and minimum rewards in different states.

Proposition 1 *For any policy $\vec{\pi}^0$:*

$$reg(\vec{\pi}^0) - cemr(\vec{\pi}^0) \leq \max_t \left[\max_s R^{*,t}(s) - \min_s R^{*,t}(s)\right] \cdot \frac{(1-\gamma^H)}{1-\gamma}$$

and

$$cemr(\vec{\pi}^0) - reg(\vec{\pi}^0) \leq \max_{t \neq H-1} \left[\max_s R^{*,t}(s) - \min_s R^{',t}(s)\right] \cdot \frac{(1-\gamma^{H-1})}{1-\gamma}$$

Proof. Detailed proof is included in appendix.
The second myopic variant of regret is called the One Step Regret (OSR).

Definition 3 *OSR for policy $\vec{\pi}^0$ is denoted by $osr(\vec{\pi}^0)$ and is defined as:*

$$osr(\vec{\pi}^0) = \min_{\hat{\vec{\pi}}^0} [v^0(\vec{\pi}^*) - v^0(\hat{\vec{\pi}}^0)] \quad \textbf{\textit{where}}$$
$$\exists \hat{t} \ s.t. \ \forall s,a,t \ \hat{\vec{\pi}}^t(s,a) = \vec{\pi}^t(s,a), \textbf{\textit{if}} \ t \neq \hat{t}$$

Intuitively, OSR for a given policy, $\vec{\pi}^0$ is the minimum regret over all the policies that are obtained by making changes to the policy in at most one time step (denoted by $\hat{t}$). $\vec{\pi}^0$ is the given policy. $\hat{\vec{\pi}}^0$ represents a policy that is different in at most one time step ($\hat{t}$) from the given policy and $\vec{\pi}^*$ is the optimal policy. While in CEMR the myopic nature is due to feasible value improvement at individual states, in OSR, the myopic nature is due to considering policy changes at most one time step.

We refer to the above variant as One Step Regret, since the policy obtained by optimizing this criterion cannot be changed in any single time step without increasing the regret. Every regret optimal policy is also an OSR optimal policy, while the converse need not be true. In section 7 we further discuss, with a concrete example, how OSR and CEMR solutions differ.

4. Uncertain MDP

A finite horizon uncertain MDP is defined as the tuple of $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, H, \alpha \rangle$. $\mathcal{S}$ denotes the set of states and $\mathcal{A}$ denotes the set of actions. $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots\}$ denotes a set of finite transition function samples, where $\mathcal{T}_k^t(s, a, s')$ denotes the probability of transitioning from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ on taking action $a \in \mathcal{A}$ at time step t according to the k^{th} element in $\mathcal{T}$. Similarly, $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \cdots\}$ is the set of finite reward function samples, where $\mathcal{R}_k^t(s, a)$ is the reward obtained on taking action a in state s at time t according to k^{th} element in $\mathcal{R}$. Finally, H is the time horizon and α is the starting state distribution.

Every element of $\mathcal{T}$ and $\mathcal{R}$ sets represent transition and reward functions respectively over the entire horizon and hence this representation captures dependence in uncertainty distributions across states. We now provide a formal definition for the independence of uncertainty distributions that is equivalent to the rectangularity property introduced in Iyengar (2005).

Definition 4 *An uncertainty distribution Δ^τ over the set of transition functions, $\mathcal{T}$ is independent over state-action pairs at various decision epochs if*

$$\Delta^\tau(\mathcal{T}) = \times_{s \in \mathcal{S}, a \in \mathcal{A}, t \leq H} \Delta_{s,a}^{\tau,t}(\mathcal{T}_{s,a}^t)$$

where $\mathcal{T} = \times_{s,a,t} \mathcal{T}_{s,a}^t$, $\mathcal{T}_{s,a}^t$ is the set of transition functions for s, a, t; $\Delta_{s,a}^{\tau,t}$ is the distribution over the set $\mathcal{T}_{s,a}^t$ and $Pr_{\Delta^\tau}(\mathcal{T}^k)$ is the probability of the transition function $\mathcal{T}^k$ given the distribution Δ^τ.

We can provide a similar definition for the independence of uncertainty distributions over the reward functions. In the following definitions, we include transition, T and reward, R functions as subscripts to indicate value (v), regret (reg) and CEMR ($cemr$) functions corresponding to a specific MDP. Existing work on computation of *maximin* policies has the following objective:

$$\pi^{maximin} = \arg\max_{\vec{\pi}^0} \min_{T \in \mathcal{T}, R \in \mathcal{R}} \sum_s \alpha(s) \cdot v_{T,R}^0(s, \vec{\pi}^0)$$

Our goal is to compute policies that minimize the maximum regret over possible functions of transitional and reward uncertainty, i.e.,

$$\pi^{reg} = \arg\min_{\vec{\pi}^0} \max_{T \in \mathcal{T}, R \in \mathcal{R}} reg_{T,R}(\vec{\pi}^0)$$

This is a challenging problem and we first explore an approximation method for minimizing maximum regret in Section 5. We then consider optimizing myopic variants of regret with an expectation that the obtained policies would yield competitive results, given their similarities to regret. Specifically, we consider the following problem in Section 6:

$$\pi^{cemr} = \arg\min_{\vec{\pi}^0} \max_{T \in \mathcal{T}, R \in \mathcal{R}} cemr_{T,R}(\vec{\pi}^0)$$

Finally, we consider the following problem in Section 7:

$$\pi^{osr} = \arg\min_{\vec{\pi}^0} \max_{T \in \mathcal{T}, R \in \mathcal{R}} osr_{T,R}(\vec{\pi}^0)$$

5. Regret Minimizing Solution

Our approach to obtaining regret minimizing solution relies on computing a solution for a sample set of model uncertainty. Specifically, we formulate the regret minimization problem for the sample set as an optimization problem with quadratic constraints and then approximate it as a Mixed Integer Linear Program (MILP). MILPs are NP-hard but software packages such as CPLEX can be used to solve them efficiently.

A sample q of model uncertainty is defined as:

$$\xi_q = \{\left\langle \mathcal{T}_q^0, \mathcal{R}_q^0 \right\rangle, \left\langle \mathcal{T}_q^1, \mathcal{R}_q^1 \right\rangle, \cdots \left\langle \mathcal{T}_q^{H-1}, \mathcal{R}_q^{H-1} \right\rangle\}$$

where $\mathcal{T}_q^t$ and $\mathcal{R}_q^t$ refer to the transition and reward function respectively at time step t in sample q . Let the set of samples be defined as ξ, which corresponds to $|\xi|$ number of discrete MDPs. Our goal is to compute *one* policy that minimizes the maximum regret over all the $|\xi|$ MDPs, i.e.,

$$\pi^{reg} = \arg\min_{\vec{\pi}^0} \max_{\xi_q \in \xi} \; [v^*_{\xi_q}(s) - \sum_s \alpha(s) \cdot v^0_{\xi_q}(s, \vec{\pi}^0) \;]$$

where $v^*_{\xi_q} = \sum_s \alpha(s) \cdot v^0_{\xi_q}(s, \vec{\pi}^*)$ and $v^0_{\xi_q}(s, \vec{\pi}^0)$ denotes the optimal expected value and the expected value for policy $\vec{\pi}^0$ respectively with respect to sample ξ_q.

Expected value for a policy $\vec{\pi}^t$, i.e., $v^t_{\xi_q}(s, \vec{\pi}^t)$ is defined as follows:

$$\begin{aligned}
v^t_{\xi_q}(s, \vec{\pi}^t) &= \sum_a \pi^t(s,a) \cdot v^t_{\xi_q}(s, a, \vec{\pi}^t), \boldsymbol{where} \\
v^t_{\xi_q}(s, a, \vec{\pi}^t) &= \mathcal{R}^t_q(s,a) + \gamma \sum_{s'} v^{t+1}_{\xi_q}(s', \vec{\pi}^{t+1}) \cdot \mathcal{T}^t_q(s,a,s')
\end{aligned}$$

The optimization problem for computing the regret minimizing policy corresponding to sample set ξ is then defined as follows:

$$\begin{aligned}
\min_{\vec{\pi}^0} \quad & reg(\vec{\pi}^0) \\
\textbf{s.t.} \quad & reg(\vec{\pi}^0) \geq v^*_{\xi_q} - \sum_s \alpha(s) \cdot v^0_{\xi_q}(s, \vec{\pi}^0) && \forall \xi_q \quad (8) \\
& v^t_{\xi_q}(s, \vec{\pi}^t) = \sum_a \pi^t(s,a) \cdot v^t_{\xi_q}(s,a,\vec{\pi}^t) && \forall s, \xi_q, t \quad (9) \\
& v^t_{\xi_q}(s,a,\vec{\pi}^t) = \mathcal{R}^t_q(s,a) + \gamma \sum_{s'} v^{t+1}_{\xi_q}(s', \vec{\pi}^{t+1}) \cdot \mathcal{T}^t_q(s,a,s') && \forall s, a, \xi_q, t \quad (10)
\end{aligned}$$

The value function in Equation (9) is a product of two variables, $\pi^t(s,a)$ and $v^t_{\xi_q}(s,a,\vec{\pi}^t)$, which hampers scalability significantly. We employ linearization and separable programming to improve scalability. The details are provided in the subsequent sections.

5.1 Mixed Integer Linear Program

The optimal policy for minimizing maximum regret in the general case is randomized. However, to account for domains which only allow for deterministic policies, we provide linearization separately for the two cases of deterministic and randomized policies.

Deterministic Policy: In case of deterministic policies, $\pi^t(s,a)$ will be a binary variable. Equation (9) contains products of binary and continuous variables. We linearize each product by defining variable $\hat{v}^t_{\xi_q}(s,a,\vec{\pi}^t) = v^t_{\xi_q}(s,a,\vec{\pi}^t) \cdot \pi^t(s,a)$ and replacing Equation (9) by the following linear constraints.

$$
\begin{aligned}
v^t_{\xi_q}(s,\vec{\pi}^t) &= \sum_a \hat{v}^t_{\xi_q}(s,a,\vec{\pi}^t) \\
\hat{v}^t_{\xi_q}(s,a,\vec{\pi}^t) &\leq v^t_{\xi_q}(s,a,\vec{\pi}^t) \\
\hat{v}^t_{\xi_q}(s,a,\vec{\pi}^t) &\leq \pi^t(s,a) \cdot M \\
\hat{v}^t_{\xi_q}(s,a,\vec{\pi}^t) &\geq v^t_{\xi_q}(s,a,\vec{\pi}^t) - (1-\pi^t(s,a)) \cdot M \qquad \forall s,a,\xi_q,t
\end{aligned} \tag{11}
$$

M is a large positive constant that is an upper bound on $v^t_{\xi_q}(s,a,\vec{\pi}^t)$. Equivalence to the product terms in Equation (9) can be verified by considering $\pi^t(s,a) = \{0,1\}$.

Randomized Policy: When $\vec{\pi}^0$ is a randomized policy, we have a product of two continuous variables. We provide a mixed integer linear approximation to address the product terms above. Let,

$$
\begin{aligned}
A^t_{\xi_q}(s,a,\vec{\pi}^t) &= \frac{v^t_{\xi_q}(s,a,\vec{\pi}^t) + \pi^t(s,a)}{2} \\
B^t_{\xi_q}(s,a,\vec{\pi}^t) &= \frac{v^t_{\xi_q}(s,a,\vec{\pi}^t) - \pi^t(s,a)}{2}
\end{aligned}
$$

Equation (9) can then be rewritten as:

$$
v^t_{\xi_q}(s,\vec{\pi}^t) = \sum_a [A^t_{\xi_q}(s,a,\vec{\pi}^t)^2 - B^t_{\xi_q}(s,a,\vec{\pi}^t)^2] \qquad \forall s,\xi_q,t \tag{12}
$$

As discussed in the next subsection on "Pruning dominated actions", we can compute upper and lower bounds for $v^t_{\xi_q}(s,a,\vec{\pi}^t)$ and hence for $A^t_{\xi_q}(s,a,\vec{\pi}^t)$ and $B^t_{\xi_q}(s,a,\vec{\pi}^t)$. We approximate the squared terms by using piecewise linear components that provide an upper bound on the squared terms. We employ a standard method from literature of dividing the variable range into multiple break points. More specifically, we divide the overall range of $A^t_{\xi_q}(s,a,\vec{\pi}^t)$ (or $B^t_{\xi_q}(s,a,\vec{\pi}^t)$), say $[br_0, br_r]$ into r intervals by using $r+1$ points, namely, $\langle br_0, br_1, \ldots, br_r \rangle$. We introduce a linear variable, $\lambda^t_{\xi_q}(s,a,w)$ for each break point w and then approximate $A^t_{\xi_q}(s,a,\vec{\pi}^t)^2$ (and $B^t_{\xi_q}(s,a,\vec{\pi}^t)^2$) as follows:

$$
A^t_{\xi_q}(s,a,\vec{\pi}^t) = \sum_w \lambda^t_{\xi_q}(s,a,w) \cdot br_w, \qquad \forall s,a,\xi_q,t \tag{13}
$$

$$
A^t_{\xi_q}(s,a,\vec{\pi}^t)^2 = \sum_w \lambda^t_{\xi_q}(s,a,w) \cdot (br_w)^2, \qquad \forall s,a,\xi_q,t \tag{14}
$$

$$
\sum_w \lambda^t_{\xi_q}(s,a,w) = 1, \qquad \forall s,a,\xi_q,t \tag{15}
$$

$$
SOS2^{s,a,t}_{\xi_q}(\{\lambda^t_{\xi_q}(s,a,w)\}_{w \leq r}), \qquad \forall s,a,\xi_q,t \tag{16}
$$

where $SOS2$ is a constraint which is associated with a set of variables of which at most two variables can be non-zero and if two variables are non-zero they must be adjacent[1].

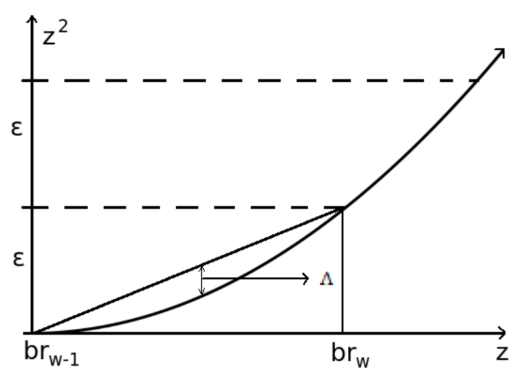
1. Details are included in appendix.

Since any number in the range lies between at most two adjacent points, we have the above constructs for the $\lambda^t_{\xi_q}(s,a,w)$ variables. We implement the above adjacency constraints on $\lambda^t_{\xi_q}(s,a,w)$ using the CPLEX Special Ordered Sets (SOS) type 2[2].

5.1.1 Approximation Error

As we are approximating the product of $\pi^t(s,a)$ and $v^t_{\xi_q}(s,a,\vec{\pi}^t)$ which in turn approximates the value of regret, it is important to provide bounds on the introduced error . We do this in three steps:

1. We first show in Proposition 2 that when approximating x^2 function using piecewise linear components, the approximation error is maximum at the midpoint of the intervals.
2. We then use the midpoint result of Proposition 2 to specifically characterise the maximum error in the approximation of $A^t_{\xi_q}(s,a,\vec{\pi}^t)^2$(and $B^t_{\xi_q}(s,a,\vec{\pi}^t)^2$) due to Equations (13)-(15) in Proposition 3. We denote the maximum error introduced in the approximation of $A^t_{\xi_q}(s,a,\vec{\pi}^t)^2$ by δ.
3. Finally, we use the result of Proposition 3 to find the error introduced in the computation of $v^t_{\xi_q}(s,\vec{\pi}^t)$ (Proposition 4) and consequently we find the bounds on regret value in Corollary 1.



(a)

Figure 1: Error associated with the interval $[br_{w-1}, br_w]$

Proposition 2 *In the approximation of z^2 function using piecewise linear components, $\lambda(w)$, the maximum approximation in any interval $[br_{w-1}, br_w]$ occurs at the mid-point.*

Proof. Without loss of generality, consider any point y in the interval $[br_{w-1}, br_w]$ (figure 1). Following Equation (13), we associate weights λ_{w-1}, λ_w with br_{w-1}, br_w respectively. We have

$$y = \lambda_{w-1} br_{w-1} + \lambda_w br_w$$

2. Our preliminary computational experiment show that using CPLEX built in SOS2 considerably improves runtime (CPLEX, 2008) compared to the formulation with binary variables.

Since we have the sum constraint in Equation (15), the above equation can be modified as:

$$y = (1 - \lambda_w) br_{w-1} + \lambda_w br_w$$
$$\implies \lambda_w = \frac{y - br_{w-1}}{br_w - br_{w-1}}$$

The error is given by the difference between Left Hand Side (LHS) and Right Hand Side (RHS) in Equation (14):

$$\Lambda = y^2 - \left[\lambda_{w-1}(br_{w-1})^2 + \lambda_w (br_w)^2\right]$$

Substituting the value of λ_w:

$$\Lambda = y^2 - \left[(br_w)^2 \cdot \frac{y - br_{w-1}}{br_w - br_{w-1}} - (br_{w-1})^2 \cdot \frac{y - br_w}{br_w - br_{w-1}}\right]$$

When Λ is maximum, we have $\frac{d\Lambda}{dy} = 0$. Therefore:

$$2y - \frac{((br_w)^2 - (br_{w-1})^2)}{(br_w - br_{w-1})} = 0$$
$$y = \frac{br_w + br_{w-1}}{2}$$

Hence proved. ■

Proposition 3 *Let [c,d] denote the range of values for $A^t_{\xi_q}(s, a, \vec{\pi}^t)$ such that $c, d > 0$ and assume we have $r+1$ points that divide $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ into r equal intervals of size $\epsilon = \frac{d^2 - c^2}{r}$ then error δ in approximating $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ satisfies: $\delta < \frac{\epsilon}{4}$.*

Proof: Let the $r+1$ points be $br_0, \ldots, br_r$. As these $r+1$ points divide $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ into equal intervals of size ϵ, we have

$$\epsilon = (br_w)^2 - (br_{w-1})^2 \quad \forall w \in [1, r]$$

Because of the convexity of x^2 function, the maximum approximation error in any interval $[br_{w-1}, br_w]$ occurs at its mid-point[3]. Hence, approximation error δ is given by:

$$\delta \leq \max_w \frac{(br_w)^2 + (br_{w-1})^2}{2} - \left[\frac{br_w + br_{w-1}}{2}\right]^2$$

where $\frac{(br_w)^2 + (br_{w-1})^2}{2}$ is the approximate value[4] of $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ at $\frac{br_w + br_{w-1}}{2}$.
Substituting $(br_w)^2 = (br_{w-1})^2 + \epsilon$

$$\delta \leq \frac{\epsilon}{4} + \max_w \frac{2 \cdot br_{w-1} \cdot (br_{w-1} - br_w)}{4}$$

3. Proposition and proof provided in Proposition 2
4. The value obtained by putting $\frac{br_w + br_{w-1}}{2}$ in the equation of line joining $(br_w, (br_w)^2)$ and $(br_{w-1}, (br_{w-1})^2)$

As $(br_{w-1} - br_w) < 0$, therefore

$$\delta < \frac{\epsilon}{4}$$

Hence proved. ■

Let $\hat{v}^t_{\xi_q}(s, \vec{\pi}^t)$ denote the approximation of $v^t_{\xi_q}(s, \vec{\pi}^t)$ obtained by using approximate values of $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ and $B^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ in Equation (12). Using the approximation error of $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ obtained by Proposition 3, we can derive bounds on the approximation of $v^0_{\xi_q}(s, \vec{\pi}^t)$.

Proposition 4

$$v^0_{\xi_q}(s, \vec{\pi}^0) - \frac{|\mathcal{A}| \cdot \epsilon \cdot (1 - \gamma^{H-1})}{4 \cdot (1 - \gamma)} \leq \hat{v}^0_{\xi_q}(s, \vec{\pi}^0) \leq v^0_{\xi_q}(s, \vec{\pi}^0) + \frac{|\mathcal{A}| \cdot \epsilon \cdot (1 - \gamma^{H-1})}{4 \cdot (1 - \gamma)}$$

Proof: From Equation (12), at time step $t+1$, the maximum approximation error in $v^{t+1}_{\xi_q}(s, \vec{\pi}^{t+1})$ is obtained when one of $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2, B^t_{\xi_q}(s, a, \vec{\pi}^t)^2$ has the maximum approximation error and the other has zero approximation error. So the maximum approximation error of $v^{t+1}_{\xi_q}(s, \vec{\pi}^{t+1})$ is given by $|\mathcal{A}| \cdot \delta$, where δ is the approximation error in $A^t_{\xi_q}(s, a, \vec{\pi}^t)^2$(or $B^t_{\xi_q}(s, a, \vec{\pi}^t)^2$) and $|\mathcal{A}|$ is the maximum number of actions across all states, all time steps.

The maximum approximation error at time step t in $v^t_{\xi_q}(s, a, \vec{\pi}^t)$ is $\gamma \cdot |\mathcal{A}| \cdot \delta$ (due to error in value function at time step $t+1$). We can combine Equation (9) and (10) as:

$$\begin{aligned} v^t_{\xi_q}(s, \vec{\pi}^t) &= \sum_a \pi^t(s, a) \cdot \left[v^t_{\xi_q}(s, a, \vec{\pi}^t) \pm \gamma \cdot |\mathcal{A}| \cdot \delta \right] \\ &= \sum_a \pi^t(s, a) \cdot v^t_{\xi_q}(s, a, \vec{\pi}^t) \pm \gamma \cdot |\mathcal{A}| \cdot \delta \end{aligned}$$

Now at time step t, the error will be $|\mathcal{A}| \cdot \delta$ plus future error from time step $t+1$ given by $\gamma \cdot |\mathcal{A}| \cdot \delta$. Extending it to $t = 0$, we will have sum of two geometric progressions, i.e.,

$$\pm \Big[|\mathcal{A}| \cdot \delta + \gamma \cdot |\mathcal{A}| \cdot \delta + \gamma^2 \cdot |\mathcal{A}| \cdot \delta ... \Big]$$

Substituting $\delta = \frac{\epsilon}{4}$ (from Proposition 3), we will have

$$v^0_{\xi_q}(s, \vec{\pi}^t) - \frac{|\mathcal{A}| \cdot \epsilon \cdot (1 - \gamma^{H-1})}{4 \cdot (1 - \gamma)} \leq \hat{v}^0_{\xi_q}(s, \vec{\pi}^t) \leq v^0_{\xi_q}(s, \vec{\pi}^t) + \frac{|\mathcal{A}| \cdot \epsilon \cdot (1 - \gamma^{H-1})}{4 \cdot (1 - \gamma)}. \quad ■$$

Corollary 1 *Let $r\hat{e}g(\vec{\pi}^0)$ denote the approximation of $reg(\vec{\pi}^0)$, then*

$$reg(\vec{\pi}^0) - \frac{|\mathcal{A}| \cdot \epsilon \cdot (1 - \gamma^{H-1})}{4 \cdot (1 - \gamma)} \leq r\hat{e}g(\vec{\pi}^0) \leq reg(\vec{\pi}^0) + \frac{|\mathcal{A}| \cdot \epsilon \cdot (1 - \gamma^{H-1})}{4 \cdot (1 - \gamma)}$$

Proof. From Equation (8) and Proposition 4, we have the proof. ■

Since the break points are fixed beforehand, we can find tighter bounds (refer to Proof of Proposition 3). Also, we can further improve the performance on both run-time and solution quality of the MILP by pruning out dominated actions and adopting sampling strategies as discussed in the next subsections.

5.1.2 Pruning Dominated Actions

We now introduce a pruning approach to remove actions that will never be assigned a positive probability in a regret minimization strategy. For every state-action pair at each time step, we define a minimum and maximum value function as follows:

$$\begin{aligned}
v^{t,min}_{\xi_q}(s,a) &= \mathcal{R}^t_q(s,a) + \gamma \sum_{s'} \mathcal{T}^t_q(s,a,s') \cdot v^{t+1,min}_{\xi_q}(s') \\
v^{t,min}_{\xi_q}(s) &= min_a \left\{ v^{t,min}_{\xi_q}(s,a) \right\} \\
v^{t,max}_{\xi_q}(s,a) &= \mathcal{R}^t_q(s,a) + \gamma \sum_{s'} \mathcal{T}^t_q(s,a,s') \cdot v^{t+1,max}_{\xi_q}(s') \\
v^{t,max}_{\xi_q}(s) &= max_a \left\{ v^{t,max}_{\xi_q}(s,a) \right\}
\end{aligned}$$

An action a' is pruned if there exists the same action a over all samples ξ_q, such that

$$v^{t,min}_{\xi_q}(s,a) \geq v^{t,max}_{\xi_q}(s,a') \ \exists a, \ \forall \xi_q$$

The above pruning step follows from the observation that an action whose best case payoff is less than the worst case payoff of another action a cannot be part of the regret optimal strategy, since we could switch from a' to a without increasing the regret value. It should be noted that an action that is not optimal for any of the samples cannot be pruned.

Algorithm 1 provides the pseudo-code for pruning step discussed earlier. At each time step, for each state we maintain an upper and lower bound for the value function. Apart from pruning, this gives us tight bounds on value function that decrease the number of break points required for linearization.

5.1.3 Greedy Sampling

The scalability of the MILP formulation is constrained by the number of samples Q. So, instead of generating only the fixed set of Q samples from the uncertainty distribution over models, we generate more than Q samples and then pick a set of size Q so that samples are "as far apart" as possible. The key intuition in selecting the samples is to consider distance among samples as being equivalent to entropy in the optimal policies for the MDPs in the samples. For each decision epoch t, each state s and action a, we define $Pr^{s,a,t}_{\xi}(\pi^{*t}_{\xi}(s,a) = 1)$ to be the probability that a is the optimal action in state s at time t. Therefore,

$$Pr^{s,a,t}_{\xi}(\pi^{*t}_{\xi}(s,a) = 1) = \frac{\sum_{\xi_q} \pi^{*t}_{\xi_q}(s,a)}{Q}$$

$$Pr^{s,a,t}_{\xi}(\pi^{*t}_{\xi}(s,a) = 0) = \frac{\sum_{\xi_q} \left(1 - \pi^{*t}_{\xi_q}(s,a)\right)}{Q}$$

Algorithm 1: PruneDominatedActions()

$t \leftarrow H - 1$
for all $\xi_q \in \xi, s \in \mathcal{S}$ **do**
  $v_{\xi_q}^{H,min}(s) \leftarrow 0$
  $v_{\xi_q}^{H,max}(s) \leftarrow 0$
while $t \geq 0$ **do**
  for all $s \in \mathcal{S}$ **do**
    for all $\xi_q \in \xi,\ a \in \mathcal{A}$ **do**
      $v_{\xi_q}^{t,min}(s,a) \leftarrow R_q^t(s,a) + \gamma \sum_{s'} \mathcal{T}_q^t(s,a,s') \cdot v_{\xi_q}^{t+1,min}(s')$
      $v_{\xi_q}^{t,max}(s,a) \leftarrow R_q^t(s,a) + \gamma \sum_{s'} \mathcal{T}_q^t(s,a,s') \cdot v_{\xi_q}^{t+1,max}(s')$
    if $\exists a' \ s.t. \ v_{\xi_q}^{t,min}(s,a') \geq v_{\xi_q}^{t,max}(s,a) \ \forall \xi_q$ **then**
      Prune a
    $v_{\xi_q}^{t+1,min}(s) = min_a v_{\xi_q}^{t,min}(s,a)$
    $v_{\xi_q}^{t+1,max}(s) = max_a v_{\xi_q}^{t,max}(s,a)$
  $t \leftarrow t - 1$

Let the total entropy of sample set ξ ($|\xi| = Q$) be represented as $\Delta S(\xi)$, then

$$\Delta S(\xi) = -\sum_{t,s,a} \sum_{z \in \{0,1\}} Pr_{\xi}^{s,a,t}(\pi_{\xi}^{*t}(s,a) = z) \cdot ln\big(Pr_{\xi}^{s,a,t}(\pi_{\xi}^{*t}(s,a) = z)\big)$$

We use a greedy strategy to select the Q samples, i.e., we iteratively add samples that maximize entropy of the sample set in that iteration.

It is possible to provide bounds on the number of samples required for a given error using the methods suggested by Shapiro (2003a). However these bounds are conservative and hence suggest a large number of samples. As we show in the experimental results section, typically when the greedy sampling is used, we only require a small number of samples and hence such bounds are not very useful in practice.

We also emphasize that, apart from greedy, it is possible to employ other procedures to generate samples. Our approaches will be applicable regardless of the sampling technique employed.

6. CEMR Minimizing Solution

The MILP based approach mentioned in the previous section can easily be adapted to minimize the maximum cumulative regret over all samples when uncertainties across states are dependent:

$$\min_{\vec{\pi}^0} \quad cemr(\vec{\pi}^0)$$

$$\textbf{s.t.} \quad cemr(\vec{\pi}^0) \geq \sum_s \alpha(s) \cdot cemr^0_{\xi_q}(s, \vec{\pi}^t), \qquad \forall \xi_q$$

$$cemr^t_{\xi_q}(s, \vec{\pi}^t) = \sum_a \pi^t(s,a) \cdot cemr^t_{\xi_q}(s,a,\vec{\pi}^t), \qquad \forall s,t,\xi_q \tag{17}$$

$$cemr^t_{\xi_q}(s,a,\vec{\pi}^t) = \mathcal{R}^{*,t}_q(s) - \mathcal{R}^t_q(s,a) + \gamma \sum_{s'} \mathcal{T}^t_q(s,a,s') \cdot cemr^{t+1}_{\xi_q}(s', \vec{\pi}^{t+1}), \quad \forall s,a,t,\xi_q \tag{18}$$

where the product term $\pi^t(s,a) \cdot cemr^t_{\xi_q}(s,a,\vec{\pi}^t)$ is approximated as described earlier.

While we were unable to exploit the independence of uncertainty distributions across states with *minimax* regret, we are able to exploit the independence with *minimax* CEMR. In fact, a key advantage of the CEMR robustness concept in the context of independent uncertainties is that it has the *optimal substructure* over time steps and hence a Dynamic Programming (DP) algorithm can be used to solve it. In the next few paragraphs and Proposition 5 we provide the proof for optimal substructure property of *minimax* CEMR.

We first precisely define the notion of samples in the case of independent uncertainties. Here, samples at each time step can be drawn independently and we introduce formal notation to account for samples drawn at each time step. Let ξ^t denote the set of samples at time step t, then $\xi = \times_{t \leq H-1} \xi^t$. Further, we use $\vec{\xi^t}$ to indicate cross product of samples from t to $H-1$, i.e., $\vec{\xi^t} = \times_{t \leq e \leq H-1} \xi^e$. Thus, $\vec{\xi^0} = \xi$. To indicate the entire horizon samples corresponding to a sample p from time step t, we have $\vec{\xi^t_p} = \xi^t_p \times \vec{\xi}^{t+1}$.

For notational compactness, we use $\Delta\mathcal{R}^{t-1}_p(s,a) = \mathcal{R}^{*,t-1}_p(s) - \mathcal{R}^{t-1}_p(s,a)$. Because of independence in uncertainties across time steps, for a sample set $\vec{\xi}^{t-1}_p = \xi^{t-1}_p \times \vec{\xi^t}$, we have the following:

$$\begin{aligned}
\max_{\vec{\xi}^{t-1}_p} cemr^{t-1}_{\vec{\xi}^{t-1}_p}(s, \vec{\pi}^{t-1}) &= \max_{\xi^{t-1}_p \times \xi^t_p} \sum_a \pi^{t-1}(s,a) \Big[\Delta\mathcal{R}^{t-1}_p(s,a) + \gamma \sum_{s'} \mathcal{T}^t_p(s,a,s') \cdot cemr^t_{\vec{\xi}^t}(s', \vec{\pi}^t) \Big] \\
&= \max_{\xi^{t-1}_p} \sum_a \pi^{t-1}(s,a) \Big[\Delta\mathcal{R}^{t-1}_p(s,a) + \gamma \sum_{s'} \mathcal{T}^t_p(s,a,s') \cdot \max_{\vec{\xi}^t_q \in \vec{\xi}^t} cemr^t_{\vec{\xi}^t_q}(s', \vec{\pi}^t) \Big]
\end{aligned} \tag{19}$$

In the following proposition, we state the optimal substructure property of minimax CEMR. Informally, it states that an optimal CEMR policy for a longer horizon includes an optimal CEMR policy for a shorter horizon problem. The optimal substructure property implies the existence of a dynamic programming algorithm and hence efficient computation of CEMR optimal policy. This is formally stated in Propostion 5 as follows:

Proposition 5 *At time step $t-1$, the CEMR corresponding to any policy π^{t-1} will have the least regret if it includes the CEMR minimizing policy from t. Formally, if $\vec{\pi}^{*,t}$ represents* *the CEMR minimizing policy from t and $\vec{\pi}^t$ represents any arbitrary policy, then:*

$$\forall s: \max_{\vec{\xi}_p^{t-1} \in \vec{\xi}^{t-1}} cemr^{t-1}_{\vec{\xi}_p^{t-1}}\Big(s, \big\langle \pi^{t-1}, \vec{\pi}^{*,t} \big\rangle\Big) \leq \max_{\vec{\xi}_p^{t-1} \in \vec{\xi}^{t-1}} cemr^{t-1}_{\vec{\xi}_p^{t-1}}\Big(s, \big\langle \pi^{t-1}, \vec{\pi}^{t} \big\rangle\Big) \tag{20}$$

$$\textbf{\textit{if}}, \quad \forall s: \max_{\vec{\xi}_q^{t} \in \vec{\xi}^{t}} cemr^{t}_{\vec{\xi}_q^{t}}(s, \vec{\pi}^{*,t}) \leq \max_{\vec{\xi}_q^{t} \in \vec{\xi}^{t}} cemr^{t}_{\vec{\xi}_q^{t}}(s, \vec{\pi}^{t}) \tag{21}$$

Proof. From Equation (19), we have:

$$cemr^{t-1}_{\vec{\xi}_p^{t-1}}\Big(s, \big\langle \pi^{t-1}, \vec{\pi}^{*,t} \big\rangle\Big) = \sum_{a \in \mathcal{A}} \pi^{t-1}(s,a)\Big[\Delta\mathcal{R}_p^{t-1}(s,a) + \sum_{s'} \mathcal{T}_p^{t-1}(s,a,s') \cdot \max_{\vec{\xi}_q^{t} \in \vec{\xi}^{t}} cemr^{t}_{\vec{\xi}_q^{t}}(s', \vec{\pi}^{*,t})\Big]$$

From Equation (21), we have:

$$\begin{aligned} cemr^{t-1}_{\vec{\xi}_p^{t-1}}\Big(s, \big\langle \pi^{t-1}, \vec{\pi}^{*,t} \big\rangle\Big) &\leq \sum_{a \in \mathcal{A}} \pi^{t-1}(s,a)\Big[\Delta\mathcal{R}_p^{t-1}(s,a) + \sum_{s'} \mathcal{T}_p^{t-1}(s,a,s') \cdot \max_{\vec{\xi}_q^{t} \in \vec{\xi}^{t}} cemr^{t}_{\vec{\xi}_q^{t}}(s', \vec{\pi}^{t})\Big] \\ &\leq cemr^{t-1}_{\vec{\xi}_p^{t-1}}\Big(s, \big\langle \pi^{t-1}, \vec{\pi}^{t} \big\rangle\Big) \end{aligned} \tag{22}$$

$$\textbf{\textit{Thus}}, \quad \max_{\xi_q \in \xi} cemr^{t-1}_{\xi_q}\Big(s, \big\langle \pi^{t-1}, \vec{\pi}^{*,t} \big\rangle\Big) \leq \max_{\xi_q \in \xi} cemr^{t-1}_{\xi_q}\Big(s, \big\langle \pi^{t-1}, \vec{\pi}^{t} \big\rangle\Big). \ \blacksquare \tag{23}$$

Extending the reasoning of Equation (22), for any policy $\vec{\pi}^0$ we have

$$\begin{aligned} &\forall s, cemr^0_{\vec{\xi}_p^0}(s, \vec{\pi}^{*,0}) \leq cemr^0_{\vec{\xi}_p^0}(s, \vec{\pi}^0) \\ \implies &\max_{\vec{\xi}_p \in \vec{\xi}^0} \sum_s \alpha(s) cemr^0_{\vec{\xi}_p^0}(s, \vec{\pi}^{*,0}) \leq \max_{\vec{\xi}_p \in \vec{\xi}^0} \sum_s \alpha(s) cemr^0_{\vec{\xi}_p^0}(s, \vec{\pi}^0) \\ \implies &cemr(\vec{\pi}^{*,0}) \leq cemr(\vec{\pi}^0) \end{aligned} \tag{24}$$

It is easy to show that minimizing CEMR also has an optimal substructure:

$$\min_{\vec{\pi}^0} \max_{\vec{\xi}_p^0} \sum_s \alpha(s) \cdot cemr^0_{\vec{\xi}_p^0}(s, \vec{\pi}^0) \implies \min_{\vec{\pi}^0} \sum_s \alpha(s) \cdot \big[\max_{\vec{\xi}_p^0} cemr^0_{\vec{\xi}_p^0}(s, \vec{\pi}^0)\big] \tag{25}$$

In Proposition 5 (extending the reasoning to $t = 1$), we have already shown that $\max_{\vec{\xi}_p^0} cemr^0_{\vec{\xi}_p^0}(s, \vec{\pi}^0)$ has an optimal substructure. Thus, Equation (25) can also exploit the optimal substructure.

MinimizeCEMR function provides the pseudo code for the DP algorithm that exploits this structure. At each stage t, we calculate the *cemr* for each state-action pair corresponding to each sample at that stage ξ^t (lines 6-9). Once all the *cemr* values are computed, we obtain the maximum *cemr* and the policy corresponding to it (line 10) by using the GetCEMR() function. In the next iteration, *cemr* computed at t is then used in the computation of *cemr* at $t-1$ using the same update step (lines 6-9).

MinimizeCEMR()

for all $t \leq H-1$ **do**
  $\xi^t \leftarrow$ GenSamples($\mathbf{T}, \mathbf{R}$)
for all $s \in \mathcal{S}$ **do**
  $cemr^H(s) \leftarrow 0$
while $t >= 0$ **do**
  for all $s \in \mathcal{S}$ **do**
    for all $\xi_q^t \in \xi^t$, $a \in \mathcal{A}$ **do**
      $cemr^t_{\xi_q^t}(s,a) \leftarrow \Delta R_q^t(s,a) +$
        $\gamma \sum_{s'} \mathcal{T}_q^t(s,a,s') \cdot cemr^{t+1}(s')$
    $\langle \pi^t, cemr^t(s) \rangle \leftarrow$ GetCEMR (s, $\{cemr^t_{\xi_q^t}(s,a)\}$)
  $t \leftarrow t-1$
return $(\vec{cemr}^0, \vec{\pi}^0)$

GetCEMR $(s, \{cemr^t_{\xi_q^t}(s,a)\})$

$\min_{\boldsymbol{\pi}} \quad cemr^t(s)$
s.t. $cemr^t(s) \geq \sum_a \pi^t(s,a) \cdot cemr^t_{\xi_q^t}(s,a), \forall \xi_q$
$\sum_a \pi^t(s,a) = 1$
$0 \leq \pi^t(s,a) \leq 1, \forall a$
return $(\pi^t, cemr^t(s))$

It can be noted that MinimizeCEMR() makes only $H \cdot |\mathcal{S}|$ calls to the LP in GetCEMR() function, each of which has only $|\mathcal{A}|$ continuous variables and at most $[1+\max_t |\xi^t|]$ number of constraints. Thus, the overall complexity of MinimizeCEMR() is polynomial in the number of samples given fixed values of other attributes.

Let $cemr^{*,H-1}(s,a)$ denote the optimal cumulative regret at time step $H-1$ for taking action a in state s and $cemr_\xi^{*,H-1}(s,a)$ denote the optimal cumulative regret over the sample set ξ. Let indicator random variable X be defined as follows:

$$X = \begin{cases} 1 & \text{if } \; cemr^{*,H-1}(s,a) - cemr_\xi^{*,H-1}(s,a) \leq \lambda \\ 0 & otherwise \end{cases}$$

By using Chernoff and Hoeffding bounds on X, it is possible to provide bounds on deviation from mean and on the number of samples at $H-1$. This can then be propagated to $H-2$ and so on. However, these bounds can be very loose and they do not exploit the properties of *cemr* functions. Bounds developed on spacings of order statistics can help exploit the properties of *cemr* functions. We will leave this for future work.

7. OSR Minimizing Solution

CEMR has two major drawbacks.

1. In case of dependent uncertainties, it provides no computational advantage over standard regret formulation as it becomes an MILP.

2. In certain problems where the reward is concentrated in a few cells, an optimal CEMR policy has a high regret value. We highlight this using the following example:

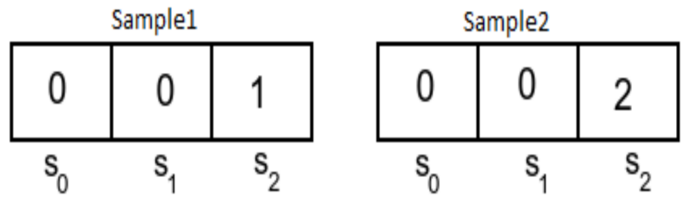


Figure 2: CEMR Example

Example 1 *Consider a 1×3 grid world as shown in Figure 2. The cells are indexed $\{s_0, s_1, s_2\}$. Consider two samples, such that the rewards in sample 1 are $(0,0,1)$ in cells s_0, s_1, s_2, respectively. We similarly define sample 2, with rewards $(0,0,2)$. Let s_0 be the start state and $H = 3$. Notice that the CEMR policy would be to stay in cell s_0 at all time steps which would give a cumulative expected myopic regret of 0, whereas the optimal regret policy is to move to cell s_2.*

CEMR policy inherently relies on "regret propagation" across states and in the case where the propagated regret value is 0, CEMR performs quite poorly. To address this concern, we introduce the One Step Regret (OSR) optimal policy that overcomes this drawback while maintaining scalability.

We first provide the main insight of the approach, which is to define the expected payoff of a policy using both frequency (dual formulation variables x) and value function (primal formulation variables v) variables (see Section 2.1), respectively. $v_{\xi_q}(\vec{\pi}^0)$ denotes the expected payoff for sample ξ_q with policy $\vec{\pi}^0$ and $\vec{x}^0_{\xi_q}$ denotes the state action frequency corresponding to policy $\vec{\pi}^0$ for sample q. We have

$$\begin{aligned} v_{\xi_q}(\alpha) &= \sum_s x^0_{\xi_q}(s) \cdot v^0_{\xi_q}(s) && (26) \\ &= \sum_s x^0_{\xi_q}(s) \cdot \sum_a \pi^0(s,a) \cdot v^0_{\xi_q}(s,a) && (27) \\ &= \sum_{t,s,a} x^t_{\xi_q}(s,a) \cdot R^t_{\xi_q}(s,a) && (28) \end{aligned}$$

Note that Equation (27) follows from Equation (26) since $v^t_{\xi_q}(s) = \sum_a \pi^t(s,a) \cdot v^t_{\xi_q}(s,a)$. From Equation (28), we can *split* the expected value computation at any time step τ as follows:

$$v_{\xi_q}(\alpha) = \underbrace{\sum_{0 \le t \le \tau-1, s, a} x^t_{\xi_q}(s,a) \cdot R^t_{\xi_q}(s,a)}_{\text{term 1}} + \underbrace{\sum_s x^\tau_{\xi_q}(s) \cdot \sum_a \pi^\tau(s,a) \cdot v^\tau_{\xi_q}(s,a)}_{\text{term 2}} \quad (29)$$

The first part in Equation (29), (labelled term 1), uses the dual variables $x^t_{\xi_q}(s,a)$ and computes the expected payoff until time step $\tau - 1$. The second part (labelled term 2) uses the primal variables $v^t_{\xi_q}(s,a)$ to compute the expected payoff from time step τ. It should be noted that the expected payoff computed using Equation (29) is equal to the value computed using Equation (26).

One Step Regret (OSR) optimizing solution is a policy where the regret value cannot be reduced by changing policy for one of the time steps. In essence, it is a local optimal solution for optimizing regret.

Before presenting the policy iteration approach, we make important observations on the approach. If we update the policy at time step τ:

- The state-action frequency variable $x^t_{\xi_q}(s,a)$ remains unchanged for all $t \in [0, \tau-1]$, since the equation (shown below) for computation of $x^t_{\xi_q}(s,a)$ does not depend on policy at τ
$$x^t_{\xi_q}(s,a) = x^t_{\xi_q}(s) \cdot \pi^t(s,a),$$
Moreover, the state occupancy variable $x^t_{\xi_q}(s)$ remains unchanged for all $t \in [0,\tau]$ since
$$x^t_{\xi_q}(s) = \sum_{s',a} x^{t-1}_{\xi_q}(s',a) \cdot \mathcal{T}^{t-1}_{\xi_q}(s',a,s),$$
- The value function variable $v^t_{\xi_q}(s,a)$ remains unchanged for all $t \in [\tau, H]$ since the expected payoff of taking an action a in state s depends on the policy in future time steps (see Equation (4)).

Consider the expected value expression as defined in Equation (29). If we start from a random policy, the observations above allow us to conclude that, on changing policy at time step τ, term 1 in Equation (29) remains unchanged. Also, the state occupancy variable $x^\tau_{\xi_q}(s)$ and value function variable $v^\tau_{\xi_q}(s,a)$ in term 2 remains unchanged. Thus we can avoid the following issues:

- The non-linearity that arises in expected payoff computation in the standard regret; and
- Recomputing the value function for all time steps $t \in [0, \tau-1]$ after each update in the policy.

Algorithm 2 provides the pseudo code for the policy iteration algorithm, which aims to compute local optimal OSR policies. We initialize π to a random policy. We then recompute the one step regret optimal policy by starting from time step $\tau = H-1$ and moving *backwards* until $\tau = 0$. After each recomputation of policy we update the value function variable for that time step (line 10). Once we recompute the policy for all time steps, we update the occupancy frequency variables for all time steps (line 16). Thus, we perform backward sweep to update the policy followed by a forward sweep to update the state occupancy variable.

The LP model in Table 1 computes the one step regret minimizing policy when a policy can only be changed at a specific time step τ where $x^{t,prev}_{\xi_q}(s,a)$, $x^{k,prev}_{\xi_q}(s)$, $v^{k,prev}_{\xi_q}(s,a)$ are state-action frequency variable, the state occupancy variable, and the value function variable, respectively, from the previous policy. We now use the OSR linear program to iteratively update the policy until it converges.

While algorithm 2 terminates when across subsequent iterations the policy remains the same, in practice, we can terminate Algorithm 2, when the difference in the regret values between any two subsequent iterations is less than some ε (line 14-15 in the algorithm).

Algorithm 2: Policy Iteration()

BOOL $\leftarrow true$
$osr^{new} \leftarrow M$ (A large constant)
$osr^{old} \leftarrow M$ (A large constant)
$\pi \leftarrow$ RandomPolicy()
while BOOL **do**
  BOOL $\leftarrow false$
  $\tau \leftarrow H - 1$
  while $\tau \geq 0$ **do**
    $\pi^{\tau,new}, osr^{new} \leftarrow$ OnestepRegret(τ)
    Update $v^{\tau}_{\xi_q}(s,a),\ v^{\tau}_{\xi_q}(s)$ using $\pi^{\tau,new}$
    if $\pi^{\tau,new} \neq \pi^{\tau}$ **then**
      BOOL $\leftarrow true$
    $\tau \leftarrow \tau - 1$
  if $osr^{old} - osr^{new} < \varepsilon$ **then**
    BOOL $\leftarrow$ false
  Update $x^{t}_{\xi_q}(s),\ x^{t}_{\xi_q}(s,a)\ \forall t \in [0, H-1]$ using $\pi^{t,new}$
  $\pi^{t} \leftarrow \pi^{t,new}\quad \forall t \in [0, H-1]$
  $osr^{old} \leftarrow osr^{new}$

$$\min_{\pi} osr \tag{30}$$

$$osr \geq v^{*}_{\xi_q} - v_{\xi_q} \qquad \forall \xi_q \tag{31}$$

$$v_{\xi_q} = \sum_{0 \leq t \leq \tau-1, s} x^{t,prev}_{\xi_q}(s,a) \cdot R^{t}_{\xi_q}(s,a) + \sum_{s} x^{\tau,prev}_{\xi_q}(s) \sum_{a} \pi^{\tau}(s,a) \cdot v^{\tau,prev}_{\xi_q}(s,a) \quad \forall \xi_q \tag{32}$$

$$\sum_{a} \pi^{\tau}(s,a) = 1 \qquad \forall s \tag{33}$$

Table 1: OnestepRegret(τ)

The value of ε is problem specific. In our experiments, we observed that the reduction in regret after the first few iterations is quite small. Therefore, early termination did not significantly affect the quality of solution.

Proposition 6 *When we have finite size sample sets taken from $\mathcal{T}$ and $\mathcal{R}$, Algorithm 2 always converges.*

Proof. Policy is changed at each iteration of Algorithm 2 only if there is a lower regret policy. Since the regret is lower bounded by zero and the algorithm improves the solution by at least ε in each iteration, we can guarantee that after a finite number of iterations, there will no reduction in regret and hence Algorithm 2 will converge. ■

Theoretically, we can only provide loose bounds on the number of iterations. Since the maximum regret is equal to the maximum expected value minus the minimum possible reward for any policy (i.e., $H * R^{min}$) over all the samples and the regret decreases at each iteration by ε, the maximum number of iterations is:

$$\frac{\max_{\xi_q}\{v^*_{\xi_q}\} - H * R^{min}}{\varepsilon}$$

where R^{min} is minimum reward over all state, action pairs, time steps and samples.

Please note that in each iteration, Algorithm 2 makes $|H|$ calls to OneStepRegret() linear program which comprises of $|S||A| + |\xi|$ variables and $2 * |\xi| + |S|$ constraints. As described in Proposition 6, the number of iterations is finite. The complexity of the linear program is polynomial with respect to the number of variables and constraints. Therefore, for a fixed number of states, actions and time horizon, the overall complexity of OSR is polynomial with respect to number of samples.

We note that the OSR solution can be extended to similarly define two step or m-step regret optimal policy. However, the expected payoff computation will no longer be linear and we will have to use linearization techniques discussed earlier for the standard regret solution.

8. Experimental Setup

In this section, we describe the two benchmark problems and the procedure employed to perform the SAA analysis.

8.1 Benchmark Problem 1: Stochastic Inventory Control under Demand Uncertainty

In the single product finite horizon stochastic inventory control problem (Puterman, 1994), at the beginning of each time period and before observing the demand, the manager determines the current inventory size and decides whether or not to order an additional stock from a supplier. Traditionally, the problem is modelled as an MDP with the assumption that the underlying demand distribution is known.

Here, we assume the demand distibution is unknown, but we have access to discrete set of demand values that have been observed in the past for each time step. We also assume that these demand value sets are independent. This allows us to model the problem as an uncertain MDP with independent uncertainties. We now define each of the uncertain MDP tuple variables as follows:

- The inventory size s^t gives us the state space.
- The order value a^t gives us the action in each state. If the inventory size is bounded by s^{max}, then the action set in state s^t is given by $\left\{0, ..., s^{max} - s^t\right\}$.
- Denote uncertain demand values at time step t by $\xi^t = \left\{d^t_0, d^t_1, ..., d^t_k\right\}$. If the unit order cost is c, the unit cost of maintaining the inventory is m and the unit revenue of selling the product is r, the reward function is given by:

$$\mathcal{R}^t_{d_q}(s,a) = r \cdot \min\left\{s^t + a^t, d^t_q\right\} - c \cdot a^t - m \cdot \max\{s^t + a^t - d^t_q, 0\}$$

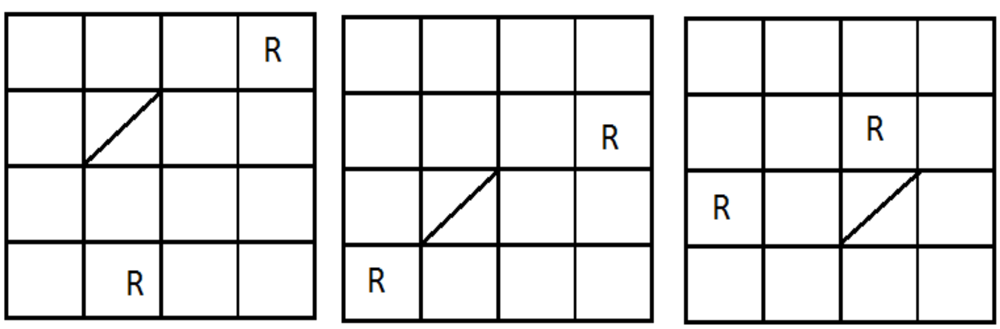
- For a fixed sample $d_q^t \in \xi^t$, the transition function is deterministic. The inventory at time step $t+1$, s^{t+1} is given by:

$$s^{t+1}(s^t, a^t) = \max\left\{s^t + a^t - d_q^t, 0\right\}$$

A standard approach for the above uncertain demand inventory management problem is to maximize the minimum expected values or *maximin solution*. In this paper, we evalutate the regret based solutions and the maximin solutions across different distributions and different cost-to-revenue ratios defined as $\frac{c+m}{r}$.

8.2 Benchmark Problem 2: Disaster Rescue

The second benchmark problem is a grid world disaster rescue problem (Bagnell et al., 2001) that is also similar to the one introduced by Nilim and Ghaoui (2005). Here again, our goal is to compute a regret based policy when there is uncertainty in both transition and reward functions. To introduce the desired uncertainty, we consider multiple maps with the same number of grids in each map but different obstacle and reward cells, i.e., we varied the cells that are labelled as obstacle or reward cells. While the stochastic obstacles lead to an uncertain transition function, stochastic reward cells lead to an uncertain reward function. For each problem, we varied the reward cells, the obstacle cells, the number of maps (characterised by the number of samples) and sizes of the grid world (characterised by the number of states). Figure 3 gives an illustration of a 4×4 grid.



(a)

Figure 3: 3 maps with different reward and obstacle cells

8.3 Sample Average Approximation (SAA) Analysis

In this section, we describe the standard method of performing the SAA analysis. As is usually the case, we employ the SAA analysis to provide posteriori bounds on the solutions obtained by our approaches and consequently also to determine the number of samples required by our algorithms to generate good quality solutions.

Each sample (scenario) is described by $i = \{i_1, i_2, i_3, ..., i_{|T|}\}$ and belong to the set I (in the case where we consider independent transition probabilities/rewards in each stage, I is the set of samples which are cross products of independent samples in each stage). Followed from the sample average approximation (SAA) method described by Shapiro (2003b), the steps to calculate the approximate optimality gap are as follows:

1. Generate the set of sample sets, $M = \{I_1, I_2, ..., I_{|M|}\}$, where each sample set is of size $|I|$. Also generate a larger sample set of size $|I'| \gg |I|$.

   - For $m = 1, ..., |M|$, solve the problem with sample set I_m to obtain the solution value $r\bar{e}g^*_m$ and policy $\bar{\pi}_m$

2. Compute the average of the objective values obtained which is a statistical lower bound of the problem and their corresponding variance as follows:

$$r\hat{e}g^* = \frac{1}{|M|} \sum_{m \in M} r\bar{e}g^*_m \text{ and } \sigma^2_{r\hat{e}g^*} = \frac{1}{|M|(|M|-1)} \sum_{m \in M} \left(r\bar{e}g^*_m - r\hat{e}g^*\right)^2.$$

3. Let $\bar{\pi}$ be the selected solution from the set of solutions obtained in Step 1. Denote by $reg^*_{I'}(\bar{\pi})$ the regret value of the policy $\bar{\pi}$ on the large sample set I'. This value is the sample average estimate of the true objective function of the policy $\bar{\pi}$. Also, its variance can be computed as follows:

$$\sigma^2_{I'}(\bar{\pi}) = \frac{1}{|I'|(|I'|-1)} \sum_{i \in I'} \left(r\bar{e}g^*_i(\bar{\pi}) - reg^*_{I'}(\bar{\pi})\right)^2$$

   where $r\bar{e}g^*_i(\bar{\pi})$ is the regret of the policy $\bar{\pi}$ corresponding to each sample $i \in I'$.

4. The absolute optimality gap of the solution $\bar{\pi}$ and its variance can be estimated as follows:

$$gap(\bar{\pi}) = |reg^*_{I'}(\bar{\pi}) - r\hat{e}g^*| \text{ and } \sigma^2_{gap}(\bar{\pi}) = \sigma^2_{I'}(\bar{\pi}) + \sigma^2_{r\hat{e}g^*}.$$

We can similarly perform the SAA analysis for MILP-CEMR.

9. Experimental Results

We now provide performance comparisonof various algorithms introduced in previous sections on the two domains, Disaster Rescue and Stochastic Inventory Control. In order to ensure the readability of the graphs, we provide names of the algorithms (along with a short description of the algorithm) in Table 2. The key performance metrics are the runtime (time taken to compute the policy) and the maximum regret value of the computed policy (i.e., the maximum regret over the set of test samples). In the graphs we refer to the maximum regret value simply as “regret”.

9.1 Performance Analysis of Sampling Strategies

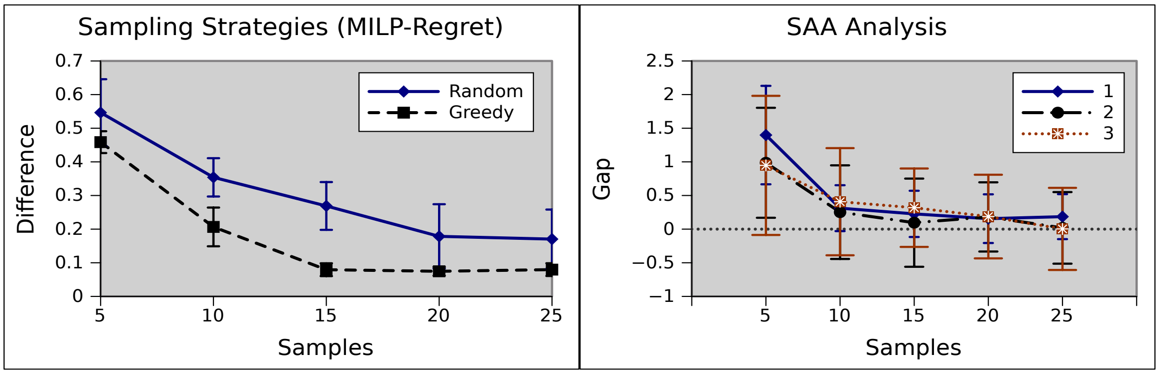
In this section, we compare the performance of using different sampling strategies and also perform SAA analysis with different number of samples.

Algorithm Name	Explanation
MILP-Regret	MILP approximation algorithm with randomized policies introduced in Section 5.1
MILP-DET-Regret	MILP approximation algorithm with deterministic policies introduced in Section 5.1
MILP-CEMR	MILP approximation algorithm for minimizing CEMR in uncertain MDPs with dependent uncertainties introduced in Section 6
OSR	Local optimal approach that minimizes one step regret introduced in Section 7
DP-CEMR	Dynamic Programming algorithm for minimizing CEMR in uncertain MDPs with independent uncertainties introduced in Section 6
Maximin	Robust algorithm that maximizes the value in the worst case

Table 2: List of algorithm names

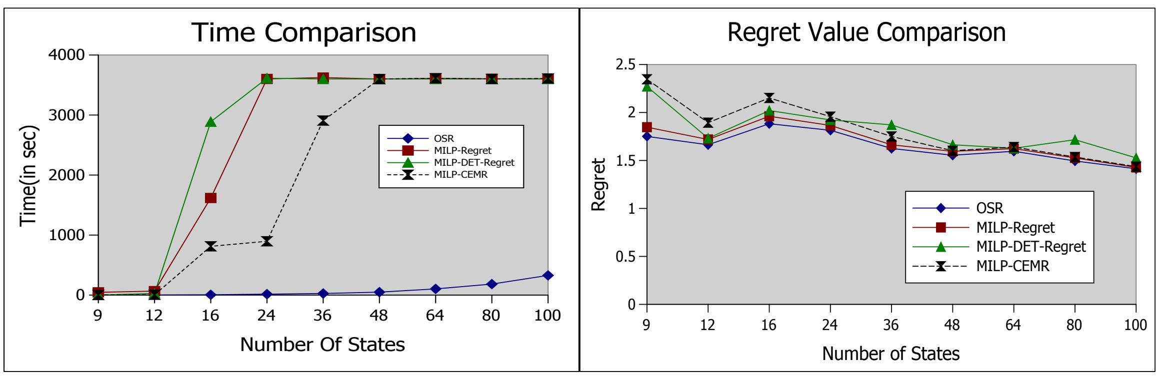
9.1.1 Greedy vs Random Sampling:

The first set of results help us determine the sampling strategy and the number of samples that are most effective for the sampling algorithms introduced in this paper. We compare the effect of using greedy sampling strategy as opposed to random sampling strategy on the MILP-Regret policy in Figure 4a. While we show the results on the disaster rescue problem (Section 8.2), the SAA analysis of the inventory management problem also yields similar results. For each grid world size, we measured the performance by varying the number of obstacles, reward cells, horizon and the number of break points employed (3-6). On X-axis, we represent the number of samples used for computation of policy (learning set). The test set from which the samples were selected consisted of 250 samples. We then obtained the policies using MILP-Regret on the learning sets generated by the two sampling strategies. On Y-axis, we show the percentage difference between maximum regret values on test and learning sample sets. We observe that for a fixed difference, the number of samples required by the greedy sampling strategy is lower in comparison to the random sampling strategy. Furthermore, the variance in difference is also lower for the greedy sampling strategy. A key result from this graph is that, even with just 15 samples, the difference with actual regret is less than 10%.



(a) (b)

Figure 4: (a) Comparison of Greedy and Random Sampling approaches (b) SAA analysis. In (a) and (b) we have 4×4 grid.



(a) (b)

Figure 5: Disaster Rescue: (a) Runtime Comparison (b) Regret value comparison of different algorithms. In (a) and (b), number of samples = 15

9.1.2 SAA Analysis And Number of Samples:

Figure 4b shows that even the gap obtained using SAA analysis is near zero (< 0.1) with 15 samples, the gap and the variance on the gap over three different settings of uncertainty labeled 1,2 and 3 are shown in the Figure 4b. Setting 3 has the highest uncertainty over the models and Setting 1 has the least uncertainty. The variance over the gap was higher for higher uncertainty settings.

Similar tests are performed for other algorithms to determine the appropriate number of samples and the appropriate sampling strategy.

Overall, as in Figures 4a and 4b, 15 samples generated using greedy sampling strategy provide stable performance and hence we use the same settings to compare the maximum regret values of the policies generated by the different sampling algorithms. Specifically, we compare the policy obtained by the four approaches with respect to the simulated regret

Number Of States	OSR	MILP -Regret	MILP-DET -Regret	MILP -CEMR
9	1.7508	1.8470	2.2710	2.3464
12	1.6635	1.7186	1.7313	1.8941
16	1.8829	1.9607	2.0203	2.1516
24	1.8145	1.8652	1.9226	1.9567
36	1.6251	1.6631	1.8710	1.7502
48	1.5544	1.5950	1.6640	1.6015
64	1.5958	1.6236	1.6270	1.6401
80	1.4947	1.5265	1.7165	1.5358
100	1.4134	1.4275	1.5267	1.4359

Table 3: Regret value comparison of different algorithms for disaster rescue problem

Number Of States	OSR	MILP -Regret	MILP-DET -Regret	MILP -CEMR
9	1.5586	45.9711	0.1202	2.0505
12	1.9231	66.0104	27.4042	4.5839
16	5.2026	1617.7615	2888.2218	812.2525
24	15.6045	3601.5521	3613.2375	896.6568
36	26.4317	3624.6838	3600.5440	2907.6537
48	49.0042	3600.3272	3600.5170	3600.1973
64	102.9541	3610.4338	3600.3104	3613.4454
80	183.2110	3605.2016	3601.5438	3600.4652
100	327.9983	3605.1479	3600.5690	3609.6117

Table 4: Runtime comparison of different algorithms for disaster rescue problem

(on 250 samples)[5] and runtime.

9.2 Performance Comparison of Regret Minimizing Algorithms

In this section, we provide a detailed comparison of the performance of different regret minimizing algorithms on the two domains described in section 8.

9.2.1 Disaster Rescue

We now provide a performance comparison of the approaches for disaster rescue domain. In Figure 5 we present the run-time and regret results obtained on disaster rescue problem for all approaches. Tables 3 and 4 provide the values shown in the Figure 5. We gradually increased the number of reward cells and obstacle cells. In the reported results, one reward cell and one obstacle cell are used for 3×3 grid and for 10×10 grid, we used 12 reward

5. As the OSR policy depends on the random policy chosen in the beginning, we ran the OSR policy computation with 10 different starting policies and the OSR policy with minimum regret is evaluated on the test set of 250 samples

cells and 3 obstacle cells. The reward values for each cell are randomly chosen between 0 and 1.

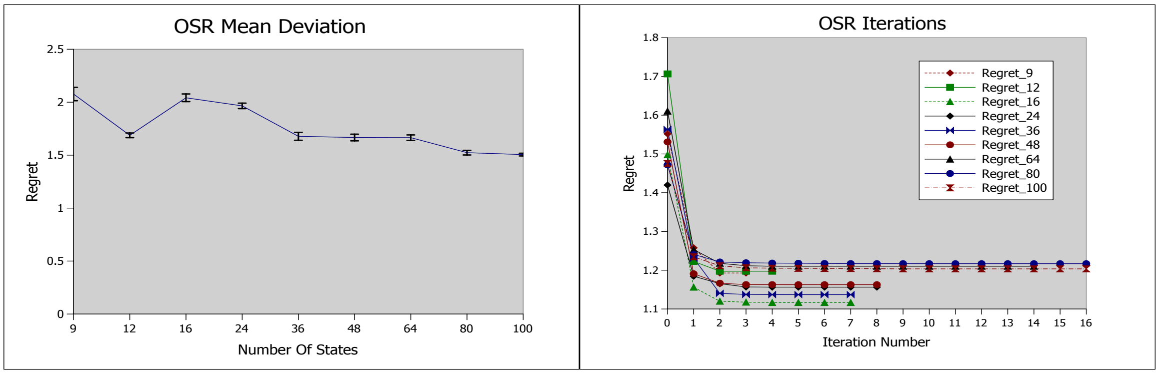
Run-time Comparison of All Approaches: In Figure 5a, we compare the time taken by the MILP-Regret, the MILP-DET-Regret, the MILP-CEMR and the OSR algorithms. On X-axis we provide the number of states (grid size) and on Y-axis we represent the time taken in seconds. As the grid size increases, the time taken by OSR is significantly lower than the time taken by MILP-Regret. For larger problem instances (the number of states > 24) MILP-Regret could not compute a solution with optimality gap $< 5\%$ within an hour but the OSR algorithm terminated within 15 minutes. For obtaining OSR policy using algorithm 2, we used the value of ε as 0.001. We later show that, using a small value of ε, does not have any impact on the quality of OSR solution.

Regret Comparison of All Approaches: Figure 5b represents the comparison of simulated regret values obtained using the four approaches. On X-axis we have the scale of the problem represented using the number of states and on Y-axis we have the simulated regret value computed over 250 samples. We observe that regret value obtained by OSR is less than or equal to the values obtained by MILP-Regret, MILP-DET-Regret and MILP-CEMR[6]. While MILP-CEMR obtained a simulated regret value within the bound provided in Proposition 1, we were unable to find any correlation in the simulated regret values of MILP-Regret and MILP-CEMR policies. We note that there is no immediate relationship between the number of states and regret values. The regret values are largely affected by the distribution of reward and obstacles cells because they induce reward and transition uncertainty. We include the results for different numbers of states for the sake of completeness and to further highlight the advantage of OSR over other approaches as increasing the number of states increases the computational complexity.

Performance Dependence of OSR on Starting Policy: Given the local improvements made by OSR, it is important to consider the effect of starting policy on the quality of solutions obtained. We computed the mean and standard deviation of simulated regret values on 250 samples obtained by OSR with 500 different starting policies. Figure 6a shows the mean and deviation of regret values for different grid sizes. It can be noted that the deviation values are very low (in the range of 10^{-2}) indicating that the quality of OSR solution is comparable to other approaches irrespective of the starting policy used in the OSR algorithm.

Convergence of OSR: To illustrate convergence with OSR, we show the regret values (on the training set containing 15 samples) obtained at each iteration of OSR. Figure 6b shows the regret value obtained for different states at each iteration. The regret values in this case are different from the values in Figure 6a as this figure shows the simulated regret values on the 250 samples test set whereas Figure 6b shows the regret values calculated on 15 samples used in one of the instance for the OSR policy computation. For these results, we executed the Algorithm 2 until the policy converges, i.e., we did not use the difference

6. For the larger problem instances where approaches were unable to compute a solution in 1 hour, we used the best solution in 1 hour



(a) (b)

Figure 6: Disaster Rescue: (a) Mean and variance of OSR with different starting policies. (b) Convergence of OSR. In (a) and (b), number of samples = 15

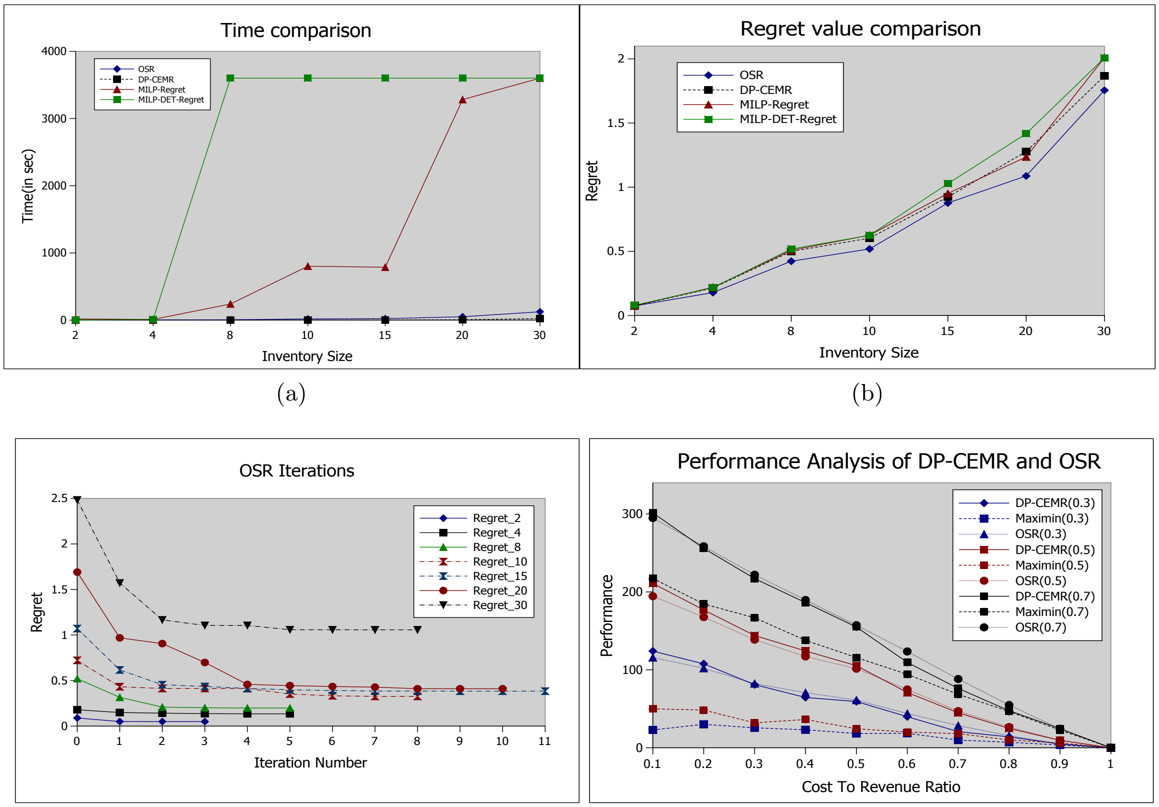
in regret values as the stopping criteria (line 14-15 in Algorithm 2).
It can be noted that, after 5-6 iterations, the change in regret value is minimal. Therefore, we can obtain a policy with local optimal regret value even if we terminate the policy computation early.

Inventory Size	**OSR**	**DP -CEMR**	**MILP -Regret**	**MILP-DET -Regret**
2	0.0745	0.0742	0.0766	0.0800
4	0.1785	0.2145	0.2200	0.2170
8	0.4230	0.4993	0.5081	0.5182
10	0.5185	0.6026	0.6257	0.6248
15	0.8767	0.9238	0.9511	1.0275
20	1.0874	1.2776	1.2356	1.4174
30	1.7555	1.8678	2.0102	2.0077

Table 5: Regret value comparison of different algorithms for inventory management problem

Inventory Size	**OSR**	**DP -CEMR**	**MILP -Regret**	**MILP-DET -Regret**
2	0.8579	0.0057	18.0950	2.3990
4	2.4898	0.0370	10.2303	9.6254
8	5.1546	0.4020	240.1578	3600.3138
10	18.4616	0.7391	802.4111	3600.4819
15	22.4234	0.9837	787.2640	3600.0820
20	50.7135	6.1107	3281.4904	3600.5303
30	124.3970	23.8553	3600.6507	3600.4732

Table 6: Runtime comparison of different algorithms for inventory management problem



(a) (b) (c) (d)

Figure 7: In (a),(b) and (c), number of samples used for policy computation is 15 and simulated regret is calculated using 250 sample test set. Cost to revenue ratio = 0.5. In (d) the maximum inventory size $(X) = 50$, $H = 20$, $|\xi^t| = 50$. Average value is computed over 10000 runs. The normal distribution mean $\mu = \{0.3, 0.4, 0.5\} \cdot X$ and $\sigma \leq \frac{min\{\mu, X-\mu\}}{3}$

.

9.2.2 Stochastic Inventory Control under Demand Uncertainty

We also conducted similar comparisons of the approaches on the well known single product finite horizon stochastic inventory control problem described in Section 8.1. As the uncertainties are independent in the inventory control problem, we can use the DP-CEMR approach to compute the cumulative regret optimizing policy.

Runtime And Regret Comparison of All Approaches: We compared the runtimes and regret values for the MILP-Regret, the MILP-DET-Regret, the DP-CEMR and the OSR approaches. Figures 7a and 7b show the comparison of these approaches for inventory control problem[7]. The demand values at each decision epoch were taken from a uniform distribution. We observed similar results as the disaster rescue domain.

7. Tables 5 and 6 contains the values used to plot these figures.

For larger inventory sizes, the MILP-Regret and the MILP-DET-Regret approaches were not able to compute a solution[8] in an hour. While DP-CEMR was the fastest amongst all approaches due to the computational complexity of the DP algorithm, the OSR also provided a comparable runtime performance. With both DP-CEMR and OSR, runtime increased linearly with the increase in the number of states. As for the regret value for this domain, the regret of the OSR policy (over 250 samples) was less than or equal to the values provided by MILP-Regret, MILP-DET-Regret and DP-CEMR.

Convergence and Dependence on Starting Policies for OSR: We observed similar behaviour with respect to convergence of the OSR approach. In all instances, OSR converged in around 10 iterations as shown in Figure 7c. With respect to dependence of final outcome on starting policy for OSR, we observed similar results as those in the disaster rescue problems. Deviation values were very low (in the range of 10^{-2}).

Comparison against Maximin: Finally, we also demonstrate the conservative nature of the Maximin policy in comparison with the policy provided by our approaches (OSR and DP-CEMR). More specifically, we compare the expected value obtained using the OSR and the DP-CEMR approaches against Maximin approach. Figure 7d provides the result of this comparison. The demand values at each decision epoch were taken from a normal distribution. We considered three different settings of mean and variance of the demands. As expected, the OSR and the DP-CEMR approaches provide much higher values than Maximin and the difference between them reduced as the cost to revenue ratio increased. It is worth noting that, although the regret of the DP-CEMR is higher than the OSR the average expected value obtained using both approaches are comparable. We obtained similar results when the demands were taken from uniform and bi-modal distributions.

10. Related Work

There are two main threads of model (transition or reward function) uncertainty considered in uncertain MDPs. The first thread assumes adversarial model uncertainty, i.e., rewards and transitions are selected in an adversarial manner (minimizing value or maximizing regret or increasing risk of failure) to the chosen policy. The second thread assumes oblivious model uncertainty, i.e., rewards and transitions are selected independently of the policy.

Our work has similarity with the work on uncertain convex programs (Calafiore & Campi, 2005) in that we also employ sampling to reduce the complexity of robust policy computation. However, due to the focus on MDPs and general models of uncertainty, there are clear differences with the work on uncertain convex programs.

10.1 Adversarial Model Uncertainty

There are three threads of research which are most relevant to the work presented in this paper on solving uncertain Markov Decision Processes with adversarial model uncertainty.

8. with less than 5% optimality gap

10.1.1 Maximin Objective

The first thread of research focuses on the well known *maximin* objective, where we compute a policy that maximizes the value in the worst case realization of the model uncertainty (Nilim & Ghaoui, 2005; Iyengar, 2005; Givan et al., 2000; Bagnell et al., 2001). This notion of robustness can be viewed as a game against the environment, where given a policy, the environment is choosing an instantiation of transition and reward functions that will minimize the expected value. Givan et al. (2000) introduced the representation of bounded parameter MDPs and provided approaches to compute pessimistic and optimistic value functions when probability for each transition can belong to an interval (and not one value). A key insight in computing the optimistic (or pessimistic) value is the definition of value maximizing (or minimizing) MDP, where uncertainty associated with each transition is instantiated to the higher (or lower) value given the constraint that sum of all outgoing probabilities for a state, action pair is 1. Bagnell et al. (2001) provide dynamic programming algorithms for *maximin* objective when uncertainty about transitions are captured as convex functions. Nilim and Ghaoui (2005) and Iyengar (2005) have identified uncertainty sets associated with transition functions involving likelihood regions or entropy bounds, where robustness can be added at practically no extra computing cost to the traditional dynamic programming approach for solving MDPs. Above mentioned work is different from the work in this paper due to not accounting for dependence in uncertainties across different states or decision epochs. Recently, techniques have been proposed to deal with dependence of uncertainties (Wiesemann et al., 2013; Mannor et al., 2012) while considering *maximin* objective.

10.1.2 Minimax Regret Objective

Due to the conservative nature of *maximin* policies (Delage & Mannor, 2010), the second thread of research pursued by Regan and Boutilier (2009) and Xu and Mannor (2009) have proposed *minimax* regret criterion (Savage, 1954) as an alternative to *maximin* objective for uncertain MDPs. Regret associated with a policy, π and an instantiation, ξ_q is defined as the difference between optimal expected value for an instantiation and expected value of π for that same instantiation. Thus, in *minimax* criterion, the goal is to find a policy that has the least value of maximum regret over all instantiations of uncertainty. This notion of robustness can be treated as a game against the environment, where the environment is choosing an instantiation of uncertainty so as to maximize the regret. In this paper, we also focus on this *minimax* notion of robustness.

While *minimax* regret policies are not conservative, computing optimal *minimax* regret policies is NP-Hard (Xu & Mannor, 2009) and hence is not scalable. Existing algorithms (Regan & Boutilier, 2010; Xu & Mannor, 2009; Regan & Boutilier, 2009) have focused on computing optimal *minimax* regret solutions to uncertain MDPs where only the reward function is uncertain. Furthermore, the uncertainties in reward for states and decision epochs are independent of each other. In this paper, we provide a general model and approaches that not only consider uncertainty over both reward and transition functions, but also consider the dependency in uncertainty across states, decision epochs.

10.1.3 Bounded Risk

The third thread of research (Chen & Bowling, 2012; Delage & Mannor, 2010) has focused on percentile measures that are based on the notions of value at risk (VaR) and conditional value at risk (CVaR). Informally, percentile measures are viewed as softer notions of robustness where the goal is to maximize the value achieved for a fixed confidence probability. Chow, Tamar, Mannor, and Pavone (2015) relate the risk sensitive MDPs to robustness and provide a novel interpretation of CVaR MDPs. They show that optimizing the CVaR of a discounted cost is equivalent to optimize the expected value of the same discounted cost in presence of worst case perturbations of transition probabilities under some budget constraints.

10.2 Oblivious Model Uncertainty

Existing work on uncertain MDPs with oblivious model uncertainty (Szörényi, Kedenburg, & Munos, 2014; Jaksch, Ortner, & Auer, 2010; Strehl & Littman, 2008; Strens, 2010; Poupart, Vlassis, Hoey, & Regan, 2006; Wang, Won, Hsu, & Lee, 2012) has focused either on:

- Identifying an unknown (and unchanging) model through Bayesian Reinforcement learning (Strens, 2010; Poupart et al., 2006; Wang et al., 2012); OR
- Identifying performance of a given learning algorithm (Szörényi et al., 2014; Jaksch et al., 2010; Strehl & Littman, 2008) while learning. There are multiple performance measures and the one of relevance to this paper is regret.

Due to the focus on oblivious model uncertainty, there is a fundamental difference to work provided in this paper.

11. Conclusions

We have introduced scalable sampling-based mechanisms for optimizing regret through the use of new regret based objectives, called CEMR and OSR, in uncertain MDPs with dependent and independent uncertainties across states. We have provided theoretical results that indicate the connection between various regret criterion, quality bounds on regret in case of MILP-Regret, optimal substructure in optimizing CEMR for independent uncertainty case and runtime performance for MinimizeCEMR. Finally, we demonstrate that the novel yet simpler OSR approach is able to consistently outperform other approaches to optimize regret on the well known inventory management problem and also on disaster rescue settings.

Acknowledgements

The research described in this paper was funded in part by the Singapore National Research Foundation (NRF) through the Singapore-MIT Alliance for Research and Technology (SMART) Center for Future Mobility (FM). It is also supported by National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in

Singapore Funding Initiative. The authors thank the anonymous referees and the editor for their valuable comments and suggestions.

Appendix A. Proof of Proposition 1

We can rewrite Equation (7) as follows:

$$cemr(\vec{\pi}^0) = v^{0,\#}(\vec{\pi}^0) - v^0(\vec{\pi}^0), \;\; \textbf{\textit{where}}$$

$$v^{0,\#}(\vec{\pi}^0) = \sum_s \alpha(s) v^{0,\#}(s, \vec{\pi}^0) \;\; \textbf{\textit{and}}$$

$$v^{t,\#}(s, \vec{\pi}^t) = \sum_a \pi^t(s,a) \cdot \Big[R^{*,t}(s) + \gamma \sum_{s'} T^t(s,a,s') \cdot v^{t+1,\#}(s', \vec{\pi}^{t+1}) \Big]$$

Therefore, we have:

$$reg(\vec{\pi}^0) - cemr(\vec{\pi}^0) = v^0(\vec{\pi}^*) - v^{0,\#}(\vec{\pi}^0)$$

The difference in the value of the optimal policy (a deterministic one) and any other policy is because of the states visited by using the policy. In the worst case for CEMR, the optimal policy visits the state with highest $R^{*,t}$ and chooses the action corresponding to $R^{*,t}$ reward. On the other hand $\vec{\pi}^0$ visits the states with the lowest $R^{*,t}$ at every time step. Therefore, we will have

$$reg(\vec{\pi}^0) - cemr(\vec{\pi}^0) \leq \sum_t \Big(\gamma^t \cdot \Big[\max_s R^{*,t}(s) - \min_s R^{*,t}(s) \Big] \Big)$$

$$reg(\vec{\pi}^0) - cemr(\vec{\pi}^0) \leq \sum_t \Big(\gamma^t \cdot \max_{t'} \Big[\max_s R^{*,t'}(s) - \min_s R^{*,t'}(s) \Big] \Big)$$

Sum of a geometric progression over the time steps yields

$$reg(\vec{\pi}^0) - cemr(\vec{\pi}^0) \leq \max_t \Big[\max_s R^{*,t}(s) - \min_s R^{*,t}(s) \Big] \cdot \frac{(1-\gamma^H)}{1-\gamma}$$

Similarly,

$$cemr(\vec{\pi}^0) - reg(\vec{\pi}^0) = v^{0,\#}(\vec{\pi}^0) - v^0(\vec{\pi}^*)$$

As $v^0(\vec{\pi}^*)$ is the value of the optimal policy, atleast for the last timestep it will have maximum value. In the worst case, for the $H-1$ steps, $\vec{\pi}^*$ will visit the state and take the action with minimum reward and $\vec{\pi}^0$ will visit the state with maximum reward. For the last timestep, both $\vec{\pi}^*$ and $\vec{\pi}^0$, visit the state with maximum reward.

$$cemr(\vec{\pi}^0) - reg(\vec{\pi}^0) \leq \sum_{t=0}^{t<H-1} \Big(\gamma^t \cdot \Big[\max_s R^{*,t}(s) - \min_s R^{',t}(s) \Big] \Big)$$

$$cemr(\vec{\pi}^0) - reg(\vec{\pi}^0) \leq \sum_{t=0}^{t<H-1} \Big(\gamma^t \cdot \max_{t' \neq H-1} \Big[\max_s R^{*,t'}(s) - \min_s R^{',t'}(s) \Big] \Big)$$

Sum of a geometric progression over the time steps yields

$$cemr(\vec{\pi}^0) - reg(\vec{\pi}^0) \leq \max_{t \neq H-1} \Big[\max_s R^{*,t}(s) - \min_s R^{',t}(s) \Big] \cdot \frac{(1-\gamma^{H-1})}{1-\gamma} \quad \blacksquare$$

Appendix B. Special Ordered Sets of Type 2 (SOS2)

A special ordered set of type 2 (SOS2) specifies an ordered set of variables. At most two variables can take non zero value and the two variables having non zero value should be adjacent to each other with respect to the order of variables. The order of variables in the set is determined by weighted value assigned to each variable in the set.

The weight associated with linear variables $\lambda^t_{\xi_q}(s,a,w)$ used in Equation (16) is br_w, i.e., the value at each break point. As only two adjacent variables in the ordered list of variables can be non-zero, this ensures that only variables corresponding to two adjacent breakpoints are non-zero. The equivalent binary integer formulation for SOS2 constraint in Equation (16), $SOS2^{s,a,t}_{\xi_q}(\{\lambda^t_{\xi_q}(s,a,w)\}_{w\leq r})$, is as follows. Let z_w be the binary variable associated with the interval $[br_{w-1}, br_w]$. We have

$$\begin{aligned}
&\sum_w \lambda^t_{\xi_q}(s,a,w) = 1\\
&\lambda^t_{\xi_q}(s,a,0) \leq z_1\\
&\lambda^t_{\xi_q}(s,a,w) \leq z_w + z_{w+1},\ \forall w \in [1, r-1]\\
&\lambda^t_{\xi_q}(s,a,r) \leq z_r\\
&\sum_{w=1}^{r} z_w = 1
\end{aligned}$$

which ensures that if $z_w = 1$, then $\lambda^t_{\xi_q}(s,a,j) = 0, \quad \forall j \neq w-1, w$.

References

Bagnell, J. A., Ng, A. Y., & Schneider, J. G. (2001). Solving uncertain Markov decision processes. Tech. rep., Carnegie Mellon University.

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*(1-2), 81–138.

Bellman, R. (1957). A Markovian decision process. *Indiana University Mathematics Journal*, *6*(4), 679–684.

Calafiore, G. C., & Campi, M. C. (2005). Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming*, *102*, 25–46.

Chen, K., & Bowling, M. (2012). Tractable objectives for robust policy optimization. In *Advances in Neural Information Processing Systems (NIPS)*.

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

CPLEX (2008). ILOG CPLEX manual. `http://ionia.bu.edu/Teaching/SE524/CPLEX11_2manuals.pdf`.

Delage, E., & Mannor, S. (2010). Percentile optimization for Markov decision processes with parameter uncertainty. *Operations Research*, *58*(1), 203–213.

Givan, R., Leach, S., & Dean, T. (2000). Bounded-parameter Markov decision processes. *Artificial Intelligence*, *122*.

Howard, R. A. (1960). *Dynamic Programming and Markov Process*. MIT Press.

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

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

Kearns, M., Mansour, Y., & Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, *49*(2-3), 193–208.

Mannor, S., Mebel, O., & Xu, H. (2012). Lightning does not strike twice: Robust MDPs with coupled uncertainty. In *International Conference on Machine Learning (ICML)*.

Mastin, A., & Jaillet, P. (2012). Loss bounds for uncertain transition probabilities in Markov decision processes. In *IEEE Annual Conference on Decision and Control (CDC), 2012*.

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

Pineau, J., Gordon, G. J., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Poupart, P., Vlassis, N. A., Hoey, J., & Regan, K. (2006). An analytic solution to discrete bayesian reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Puterman, M. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons.

Regan, K., & Boutilier, C. (2009). Regret-based reward elicitation for Markov decision processes. In *Uncertainty in Artificial Intelligence (UAI)*.

Regan, K., & Boutilier, C. (2010). Robust policy computation in reward-uncertain MDPs using nondominated policies. In *National Conference on Artificial Intelligence (AAAI)*.

Savage, L. (1954). *The Foundations of Statistics*. Wiley.

Shapiro, A. (2003a). Monte carlo sampling methods. In *Stochastic Programming*, Vol. 10 of *Handbooks in Operations Research and Management Science*. Elsevier.

Shapiro, A. (2003b). Monte carlo sampling methods. In Ruszczynski, A., & Shapiro, A. (Eds.), *Stochastic Programming*, Vol. 10 of *Handbooks in Operations Research and Management Science*. Elsevier.

Strehl, A. L., & Littman, M. L. (2008). An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, *74*(8), 1309–1331.

Strens, M. (2010). A bayesian framework for reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Szörényi, B., Kedenburg, G., & Munos, R. (2014). Optimistic planning in Markov decision processes using a generative model. In *Advances in Neural Information Processing Systems (NIPS)*.

Wang, Y., Won, K. S., Hsu, D., & Lee, W. S. (2012). Monte carlo bayesian reinforcement learning. In *International Conference on Machine Learning (ICML)*.

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

Xu, H., & Mannor, S. (2009). Parametric regret in uncertain Markov decision processes. In *IEEE Conference on Decision and Control (CDC)*.