Regret-based Reward Elicitation for Markov Decision Processes

Kevin Regan and **Craig Boutilier**
Department of Computer Science
University of Toronto
{kmregan, cebly}@cs.toronto.edu

Abstract

Traditional methods for finding optimal policies in stochastic, multi-step decision environments require a precise model of both the environment dynamics and the rewards associated with taking actions and the effects of those actions. While dynamics are often easily learnable through observation—and in many domains are stable across different users—reward functions are more problematic. In practice, it is often cognitively complex and time-consuming for users to precisely specify rewards. This work casts the problem of specifying rewards as one of preference elicitation. We will first discuss how robust policies can be computed for Markov Decision Processes given partial reward information using the minimax regret criterion. We will then show how regret can be reduced by efficiently eliciting rewards information using bound queries. Regret-based elicitation of reward offers an efficient way to produce desirable policies without resorting to the precise specification of the entire reward function, and as such, opens up a new avenue for the design of stochastic controllers.

1 Introduction

Markov decision processes (MDPs) have proven to be an extremely useful formalism for decision making in stochastic environments. However, the specification of an MDP by a user or domain expert is difficult (e.g., cognitively demanding) and time consuming. Much work has been devoted to learning the dynamics of stochastic systems from transition data, both in offline and online (i.e., reinforcement learning) settings (e.g., (Sutton & Barto 1998)); and even when specified manually by, say a domain expert, in many AI applications dynamics are stable across entire user populations.

The specification of reward functions for MDPs is much more problematic. Unless intelligent agents are designed for single users, it is impossible to specify a reward function *a priori* at design time: the reward function must reflect the preferences of the user on whose behalf the agent is acting. Even online RL methods require the specification of a user's reward function in some form (unlike state variables, it is impossible to directly observe a reward function, a convenient fiction almost always assumed in the RL literature).

Reward specification is difficult for two key reasons. First it requires the translation of user preferences—general views of what states are "good" and "bad"—into a precise numerical reward for states and actions of an MDP. As has been well-recognized in decision analysis for decades, people find it extremely difficult to quantify their strength of preferences precisely using utility functions (and, by extension, reward functions) (French 1986). The problem is further exacerbated by the potential conflation of immediate reward (i.e., $r(s,a)$) with long-term value (either $Q(s,a)$ or $V(s)$): states can be viewed as good or bad based on their ability to make other good states reachable.

It has been recognized in the literature on preference elicition that full specification of a utility function is not usually needed to make optimal or near-optimal decisions (Chajewska, Koller, & Parr 2000; Boutilier *et al.* 2006). Interactive elicitation and optimization techniques take advantage of the outcome dynamics and any previous utility information gleaned to direct further elicitation effort to those parts of a utility function that will be most useful in guiding a decision. Similarly, in an MDP setting, a near-optimal policy can be often found with a very partial specification of the reward function. For instance, bounds on the reward function in conjunction with the transition dynamics of the MDP can render certain regions of state space provably dominated by others (w.r.t. value).

In this paper, we adopt a preference elicitation perspective on specification of MDP reward functions. Our approach can be viewed as one of interactive optimization. Our work is divided into two main components. Given specific bounds on the space of feasible reward functions, we first develop the *minimax regret* decision criterion (Boutilier *et al.* 2006) for MDPs: intuitively, this determines the policy that has minimum regret, or loss w.r.t. the optimal policy, over all possible reward function realizations. Unlike other work on robust optimization for MDPs, which focuses on the maximin decision criterion (Bagnell, Ng, & Schneider 2001; Nilim & Ghaoui 2004; Iyengar 2005; McMahan, Gordon, & Blum 2003), we will see that minimax regret determines policies that with superior performance in the presence of reward function uncertainty. We then develop a simple elicition procedure that exploits the information provided by the minimax-regret solution to guide the querying process. We focus on simple schemes that refine the upper and lower bounds of specific reward values. We show that good or optimal policies can be determined with very imprecise reward functions when elicitation effort is focused in this way. Our work thus tackles the problem of reward function precision directly. While we do not address the issue of reward-value conflation in this model, we will discuss it further below.

We focus on settings in which elicitation occurs offline and queries can be posed regarding any aspect of the reward function. This stands in contrast to the typical online RL formalisms, in which the RL agent is situated in state-space and gathers local information about reward (and transition dynamics) for the current state and action only. As such, no exploration-exploitation tradeoff arises in our model.

We proceed by first reviewing some basic notation for MDPs and define minimax regret for imprecise-reward MDPs in Section 2. We then describe an iterative constraint generation procedure used to compute minimax regret in Section 3. Section 4 discusses how queries are used to reduce regret and Section 5 analyses the efficiency of the regret computation and elicitation procedure. We show that minimax regret provides better policies for imprecise-reward MDPs than alternative robust optimization criteria like maximin, and suggests better queries as well. Section 6 discusses related work in detail. We offer some conclusions and directions for future work in Section 7.

2 Notation and Problem Formulation

In this section we review some notation and properties of Markov Decision Properties and the minimax regret criterion.

2.1 Markov Decision Processes

We restrict our discussion to the infinite horizon MDP definite by the tuple $\langle S, A, \{P_{sa}\}, \gamma, \alpha, r \rangle$, where

S is a finite set of N **states**.

A is a finite set of k **actions**.

$P_{sa}(\cdot)$ are the state **transition probabilities** upon taking action a in state s.

γ is the **discount factor**

$\alpha(\cdot)$ is the **initial state distribution**.

$r : S \times A \rightarrow \mathbb{R}$ is the **reward function**.

For convenience we express functions with matrix-vector notation using a bold font: e.g., $\mathbf{r}$ is a vector of length $N \times k$ whose entries are rewards $r(s, a)$; $\mathbf{P}$ is the $N \times k \times N$ transition matrix. Let $\mathbf{r_a}$ denote the N-vector obtained by restricting $\mathbf{r}$ to action a, and similarly define $N \times N$ matrix $\mathbf{P_a}$. We define the matrix $\mathbf{E}$ follows: $\mathbf{E}$ has one row for each state-action pair and one column per state. The entry for row sa and column s' contains $P_s a(s')$ when $s' \neq s$, and $P_s a(s') - 1$ when $s' = s$.[1]

A deterministic **policy** is a mapping $\pi : S \rightarrow A$ and the **value function** for π is given by

$$V^\pi(s_0) = E\left[\sum_{i=0}^{\infty} \gamma^i r(s_i, \pi(s_i)) \,\middle|\, \pi \right]$$

where the policy π, starting in state s_0, induces a distribution over the sequence $\{s_0, s_1, \dots\}$ of visited states. The value function satifies:

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^\pi(s')$$

or equivalently in vector notation (and a slight abuse of subscript):

$$\mathbf{V}^\pi = \mathbf{r}_{\mathbf{a}_\pi} + \gamma \mathbf{P}_{\mathbf{a}_\pi} \mathbf{V}^\pi \tag{1}$$

We also define the **Q-function** $Q : S \times A \rightarrow \mathbb{R}$ as:

$$\mathbf{Q}^\pi_\mathbf{a} = \mathbf{r_a} + \gamma \mathbf{P_a} \mathbf{V}^\pi,$$

i.e., the value of executing π forward after taking action a.

A policy can also be naturally represented using the **visitation frequency** function $f : S \times A \rightarrow \mathbb{R}$ which expresses the total discounted joint probability of being in a state and taking an action. A visitation frequency f lies in the valid set of visitation frequencies $\mathcal{F}$ (with respect to the transition dynamics) if obeys the following relationship (Puterman 1994)

$$\gamma \mathbf{E} \mathbf{f} + \alpha = \mathbf{0}. \tag{2}$$

This simply requires that the sum of the frequencies into a state to be equal to the sum of the frequencies out. Policy π induces a unique visitation frequency vector $\mathbf{f}^\pi$, satisfying the constraint that $\pi(s, a) = \mathbf{f}^\pi_\mathbf{sa} / (\sum \mathbf{a}' \mathbf{f}^\pi_{\mathbf{sa}'})$ in the solution of Eq. 2. (For deterministic policies, this is equivalent to requiring that $\mathbf{f}^\pi_\mathbf{sa} = \mathbf{0}$ for all a other than $\pi(s)$.)

The **optimal value function** V^* is $\sup_\pi \alpha \mathbf{V}^\pi$. It must satisfy the condition that

$$\alpha \mathbf{V}^* = \mathbf{r}^\top \mathbf{f}^* \tag{3}$$

where $\mathbf{f}^* = \sup_\mathbf{f} \mathbf{r}^\top \mathbf{f}$ (Puterman 1994). Thus, an determining an optimal policy is equivalent to finding the optimal visitation frequencies $\mathbf{f}^*$.

2.2 Minimax Regret

The difficulty of specifying reward functions means that we will often be faced with computing policies with imprecisely specified rewards. Formally we assume only that $\mathbf{r} \in \mathcal{R}$, where $\mathcal{R}$ denotes the set of feasible reward

[1]This allows for $\mathbf{P_a V} - \mathbf{V} + \mathbf{r_a} = \mathbf{0}$ to be replaced with the simpler $\mathbf{EV} + \mathbf{r} = \mathbf{0}$.

functions. This could be dictated by a set of bounds on plausible reward values specified *a priori* by a user or domain expert; it could constraints that emerge from the result of an elicitation process (as discussed below); or could arise from observations of behavior (as in inverse RL (Ng & Russell 2000)). Even in the latter situations, we are unlikely to gave full reward information. Thus we require a criterion by which to compare policies with an imprecise-reward MDP.

We adopt the *minimax regret criterion*, one that has been used with some success in nonsequential decision problems (Boutilier, Sandholm, & Shields 2004; Boutilier *et al.* 2006). Let $\mathcal{R}$ be the set of feasible reward functions. Minimax regret can be defined in three stages (and can be viewed as a game between a decision maker and an adversary). Let $\mathbf{r}$ be some reward function and $\mathbf{f} \in \mathcal{F}$ be a vector of valid visitation frequenices (corresponding to a policy chosen by the decision maker). The *regret* of $\mathbf{f}$ w.r.t. $\mathbf{r}$ is:

$$\text{Regret}(\mathbf{f},\mathbf{r}) = \max_{\mathbf{g}\in\mathcal{F}} \quad \mathbf{r}\cdot\mathbf{g} - \mathbf{r}\cdot\mathbf{f}$$

This reflects the loss associated with $\mathbf{f}$ relative to acting optimally should the true reward function be $\mathbf{r}$. Since the true reward is unknown, we allow an adversary to choose any reward $\mathbf{r} \in \mathcal{R}$ to maximize the regret of the decision maker. Define **max regret** of f w.r.t. feasible reward set $\mathcal{R}$ to be:

$$\text{MaxRegret}(\mathbf{f},\mathcal{R}) = \max_{\mathbf{r}\in\mathcal{R}} \quad \text{Regret}(\mathbf{f},\mathbf{r})$$

Finally, we wish to minimize the maximum regret that can be dictated by the adversary. Define the **minimax regret** of feasible reward set $\mathcal{R}$ to be:

$$\text{MinimaxRegret}(\mathcal{R}) = \min_{\mathbf{f}\in\mathcal{F}} \quad \text{MaxRegret}(\mathbf{f},\mathcal{R})$$

The minimax regret criterion has a variety of desirable properties relative to other robust decision criteria. Compared to Bayesian methods that compute expected value using a prior over $\mathcal{R}$ (Chajewska, Koller, & Parr 2000; Boutilier 2002), minimax regret provides worst-case bounds on loss Specifically, let $\mathbf{f}$ be the minimax regret optimal visitation frequencies and let δ be the max regret achieved by $\mathbf{f}$; then, given any instantiation of $\mathbf{r}$, no policy has expected value greater than δ more than that of $\mathbf{f}$. Regret can often be much easier to compute to expected value given a prior density over $\mathcal{R}$.

Other work on robust optimization (Bagnell, Ng, & Schneider 2001; McMahan, Gordon, & Blum 2003; Nilim & Ghaoui 2004; Iyengar 2005) uses the maximin criterion which is defined for reward as follows:

$$\text{Maximin}(\mathcal{R}) = \max_{\mathbf{f}\in\mathcal{F}} \min_{\mathbf{r}\in\mathcal{R}} \quad \mathbf{r}\cdot\mathbf{f} \tag{4}$$

The **maximin** criterion leads to conservative policies by optimizing against the worst possible instantiation of $\mathbf{r}$ (as we will see).

We argue that the minimax regret criterion provides a less conservative and more intuitive measure by assessing the policy *ex post* and making comparisons with respect to a realized reward. Thus, the policy π would only be penalized on reward r if there existed a policy π' that did better with respect to that reward. McMahan, Gordon, and Blum (2003) developed a linear programming approach to efficiently computing the maximin value of an MDP. We will compare this approach to our minimax regret model empirically below.

3 Minimax Regret Computation

A variety of methods have been developed for computing robust policies with respect to uncertainty over transition and reward functions. Specifically, for the maximin criterion, several (related) dynamic programming approaches, which decomposed the problem across time, have been proposed (Bagnell, Ng, & Schneider 2001; Nilim & Ghaoui 2004; Iyengar 2005). Such a dynamic programming decomposition does not appear tenable for minimax regret since it grants the adversary too much power by allowing the rewards to be set independently at each time step. Following the formulations for nonsequential problems developed in (Boutilier, Sandholm, & Shields 2004; Boutilier *et al.* 2006), we instead formulate the optimization using a series of linear and mixed integer programs that enforce a consistent choice of reward function across time.

Assume a feasible reward set $\mathcal{R}$ given by a convex polytope $\mathbf{Cr} \le \mathbf{d}$. The constraints on $\mathbf{r}$ arise as discussed above (prior bounds, elicitation, or behavioral observation). Minimax regret can then be expressed as following optimization:

$$\begin{aligned} \min_{\mathbf{f}} \quad & \max_{\mathbf{g}} \max_{\mathbf{r}} \quad \mathbf{r}\cdot\mathbf{g} - \mathbf{r}\cdot\mathbf{f} \\ \text{subject to:} \quad & \gamma\mathbf{E}^\top\mathbf{f} + \alpha = \mathbf{0} \\ & \gamma\mathbf{E}^\top\mathbf{g} + \alpha = \mathbf{0} \\ & \mathbf{Cr} \le \mathbf{d} \end{aligned}$$

This is equivalent to the simple minimization:

$$\begin{aligned} \underset{\mathbf{f},\delta}{\text{minimize}} \quad & \delta \\ \text{subject to:} \quad & \mathbf{r}\cdot\mathbf{g} - \mathbf{r}\cdot\mathbf{f} \le \delta \quad \forall\, \mathbf{g}\in\mathcal{F}, \mathbf{r}\in\mathcal{R} \\ & \gamma\mathbf{E}\mathbf{f} + \alpha = \mathbf{0} \end{aligned} \tag{5}$$

Rather than enumerate what is essentially an infinite number of constraints [2], we use Benders Decomposition (Benders 1962) to iterative generate constraints.

Benders Decomposition employs a convergent series of approximations to program (5). At each iteration two optimizations are solved: The **master problem** initially takes the form of the original formulation without constraints. The **subproblem** takes the current solution to the master problem and generates the maximally violated constraint in the master problem. The maximally violated constraint is then added to the master problem and this process iterates until the solution to the master problem and subproblem converge. In the context of our game, the choice of policy by the decision

[2]Since both $\mathbf{r}$ and $\mathbf{g}$ take on values from the reals.

maker maps to the solution to the master problem, and the max regret choice of policy and reward function by the adversary maps to the solution to the subproblem at each time step. At a high level the iterative constraint generation procedure can be thought of as executing the following algorithm:

```
while $\Delta > \epsilon$ :
    minimax regret, $\mathbf{f} :=$ master()
    max regret, $\langle \mathbf{g}, \mathbf{r} \rangle :=$ subproblem($\mathbf{f}$)
    add constraint for $\langle \mathbf{g}, \mathbf{r} \rangle$ to master
    $\Delta =$ max regret $-$ minimax regret
```

Formally, the master problem is formulated as:

$$\begin{aligned} \min_{\mathbf{f},\delta} \quad & \delta \\ \text{subject to:} \quad & \mathbf{r}_i \cdot \mathbf{g} - \mathbf{r_i} \cdot \mathbf{f} \qquad \forall \, \langle \mathbf{g_i}, \mathbf{r_i} \rangle \in \text{GEN} \\ & \gamma \mathbf{E} \mathbf{f} + \alpha = \mathbf{0} \end{aligned}$$

Each iteration i of constraint generation solves the subproblem and adds the pair $\langle \mathbf{g}_i, \mathbf{r}_i \rangle$ to GEN, the set of generated constraints.

To compute the maximally violated constraint, the subproblem uses the current solution $\mathbf{f}$ and computes the policy and reward function that maximizes regret of $\mathbf{f}$. The maximization is specified as a mixed integer program, using value and Q-functions:[3]

$$\begin{aligned} \underset{\mathbf{Q},\mathbf{V},\mathbf{I},\mathbf{r}}{\text{maximize}} \quad & \alpha \cdot \mathbf{V} - \mathbf{r} \cdot \mathbf{f} & & (6) \\ \text{subject to:} \quad & \mathbf{Q_a} = \mathbf{r_a} + \gamma \mathbf{P_a} \mathbf{V} & \forall \, a \in A & \\ & \mathbf{V} \geq \mathbf{Q_a} & \forall \, a \in A & \quad (7) \\ & \mathbf{V} \leq (\mathbf{1} - \mathbf{I_a}) \mathbf{M_a} + \mathbf{Q_a} & \forall \, a \in A & \quad (8) \\ & \mathbf{C} \mathbf{r} \leq \mathbf{d} & & \\ & \sum_a \mathbf{I_a} = \mathbf{1} & & \\ & \mathbf{I_a} \in \{\mathbf{0}, \mathbf{1}\} & & \\ & \mathbf{M_a} = \mathbf{M_a^\top} - \mathbf{M_a^\perp} & & \end{aligned}$$

where the constraints (7) and (8) use the Big-M trick to ensure that the optimal value $V(s) = Q(s,a)$ for a single action a. We ensure a tight M by setting $\mathbf{M_a^\top}$ to be the Q-value $\mathbf{Q_a^\top}$ of the optimal policy with respect to best setting of rewards and $\mathbf{M_a^\perp}$ to be the Q-value $\mathbf{Q_a^\perp}$ of the optimal policy with respect to best worst of rewards.

The subproblem does not directly produce a pair $\langle \mathbf{g}_i, \mathbf{r}_i \rangle$ to be added to the master constraint set; instead it provides $\mathbf{r}_i$ and $\mathbf{V}_i$. However, we do not need access to $\mathbf{g}_i$ directly; the constraint can be posted using the reward function $\mathbf{r}_i$ and the value $\alpha \cdot \mathbf{V}_i$, since $\alpha \cdot \mathbf{V}_i = \mathbf{r}_i \cdot \mathbf{g}_i$ (and $\mathbf{g}_i$ is required to determine this adversarial value in the posted contraint).

In practice we have found that the iterative constraint generation procedure converges extremely quickly. Figure 3 shows each step of the procedure for an MDP with 10 states and 5 actions. Section 5 provides further analysis of the efficiency of the minimax regret computation.

[3]Specifying max regret in terms of visitation frequencies results in a non-convex quadratic program).

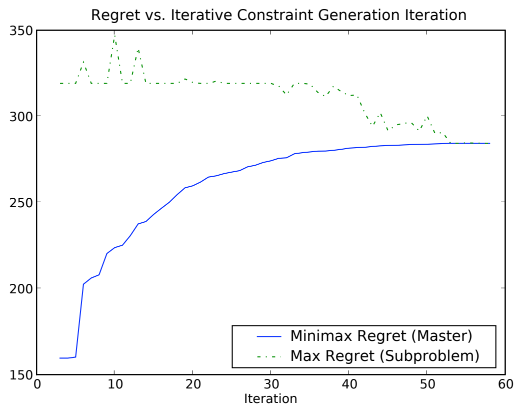

Figure 1: Convergence of Iterative Constraint Generation Procedure

4 Reward Elicitation

Reward elicitation and assessment can proceed in a variety of ways. Many different query forms can be used for user interaction, including questions that ask for comparisons of immediate reward or value or states or state-action pairs; comparisons of trajectories through state space; or full or partial policy comparisons. Similarly, observed user behavior can be used to induce constraints on the reward function under assumptions of user "optimality" (Ng & Russell 2000). In this work, we focus on simple *bound queries*, though our strategies can be adapted to more general query types.

We assume that $\mathcal{R}$ is given by a set of upper and lower bounds, one pair of bounds on $r(s,a)$ for each state state-action pair. A bound query takes the form *"Is $r(s,a) \geq b$?"* where b lies between the upper and lower bound on $r(s,a)$. While this query appears to require a quantitative assessment on direct value/reward on the part of a user, it can be rephrased in terms of a *standard gamble* (Keeney & Raiffa 1976), a standard device used in decision analysis to reduce this to preference query over two outcomes (on of which is stochastic). For simplicity, we simply write it in this bound form. Bound queries offer a natural starting point since require a minimal amount of cognitive effort to answer. However, other queries are likely to prove equally, if not more, effective.

As with the types of possible queries, there are many ways to select the point (s,a) at which to ask the query. We explore some simple myopic heuristic criteria that are very easy to compute (these are based on criteria suggested in (Boutilier *et al.* 2006)). We note that a myopically optimal query strategy, one that asks the query with the potential to maximally reduce regret, requires the solution of multiple minimax optimizations, one each answer to each possible query. This motivates the need for simple heuristics.

The first selection heuristic is called **halve largest gap (HLG)**, which selects the point (s, a) with the largest gap between the upper and lower bound. Formally, we define the gap $\Delta(s, a)$ by:

$$\Delta(s,a) = \max_{r' \in \mathcal{R}} r'(s,a) - \min_{r \in \mathcal{R}} r(s,a)$$

and the point (s^*, a^*) with the largest gap as:

$$\operatorname*{argmax}_{a^* \in A, s^* \in S} \Delta(s^*, a^*)$$

The second selection heuristic called **current solution** uses the minimax optimal visitation frequencies $\mathbf{f}$ to weight each gap. Formally it selects the point (s^*, a^*)

$$\operatorname*{argmax}_{a^* \in A} \operatorname*{argmax}_{s^* \in S} f(s^*, a^*)\Delta(s^*, a^*)$$

The intuition here is that the reward function at state-action points that are frequently visited has a high impact on the minimax regret. Once the point (s^*, a^*) is selected, the midpoint of the interval $r(s^*, a^*)$. is chosen as the bound b, thus either response will reduce the size of the interval by half.[4] It is easy to apply the current solution heuristic to the maximin criterion as well, using the visitation frequencies associated with the maximim policy.

Note that should we start with upper and lower bounds on each $r(s, a)$, the initial polytope $\mathcal{R}$ is hyperrectangular. Bound queries preserve this hyperrectangular form. With these upper and lower bounds, the minimax regret IP formulation can be considerably simplified using techniques described in (Boutilier *et al.* 2006), which can lead to considerable computational speed up. However, we rely on the more general formulation here.

5 Experiments

We evaluate the performance of our minimax regret approach in two ways. First we assess the scalability of our computational formulation. Second, we examine the effectiveness of minimax regret as a driver of elicitation. Specifically, we are interested in how close to optimal the policies generated are as a function of the number of queries used to reduce reward function uncertainty.

We used a set of random generated MDPs to evaluate our approach. We used a hyper-rectangle to represent the set of feasible reward functions for each MDP, where the reward for each s, a-pair is bounded independently. The upper and lower bounds for the interval are drawn from uniform distribution $U[0, 10]$, and the true reward was then uniformly drawn to lie between those bounds. (True reward is used to simulate the response to queries and measure the quality of the robust policies suggested by minimax regret and maximin.)

We imposed some structure on the MDP by creating a semi-sparse transition function. From any state s we restrict transitions to only reach $\lceil \log N \rceil$ other states. For each state s and action a we drawn the possible reachable states from a uniform distribution and place a Gaussian distribution over the transition function to these states. We used a uniform initial state distribution α and a discount factor $\gamma = 0.95$.

CPLEX 11 was used as the solver for all of the mixed integer and linear programs all code was run on a Dell PowerEdge 2950 server with dual quad-core Intel E5355 CPUs. In each experiment 20 MDPs were randomly generated and the results were averaged.

5.1 Computational Efficiency

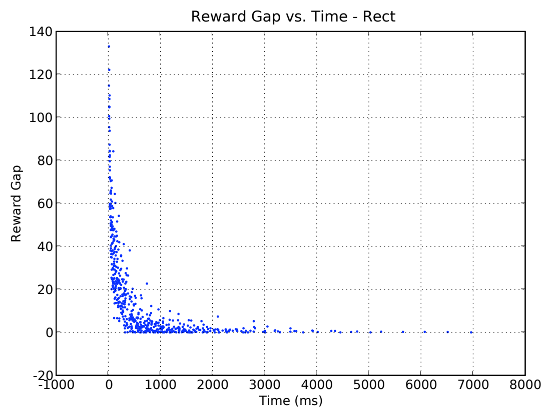

Figure 2: Drop in Regret Gap over during ICG

To measure the performance of minimax regret computation, we first examined the iterative constraint generation procedure on. For each step in the procedure, Figure 2 plots the gap between the master problem and the subproblem at each iteration against the time (in msec) to reach that iteration. Results are shown for 20 randomly generated MDPs with ten states and five actions (scatterplot shows each iteration for each MDP).

Figure 3 shows how the time to compute minimax regret increases with the size of the MDP. Here we fixed the number of actions at 5 and varied the number of states. It is clear that the iterative generation approach developed in Section 3 scales superlinearly and computing exact minimax regret is only feasible for small MDPs using this formulation.[5] On the other hand, min-

[4]We can also use the visitation frequencies $\mathbf{g}$ in the adversarial policy, and choose the maximal weighted gap using both $\mathbf{f}$ and $\mathbf{g}$. This too can help reduce regret by limiting the adversary's ability to improve on policy $\mathbf{f}$. However, since maximin has no such notion of adversary (see below), we limit attention to $\mathbf{f}$ to allow a fairer comparison.

[5]Of note, the computations shown here are using the initial reward uncertainty. As queries refine the reward polytope, regret computation becomes faster in general. This has positive implications for anytime computation.

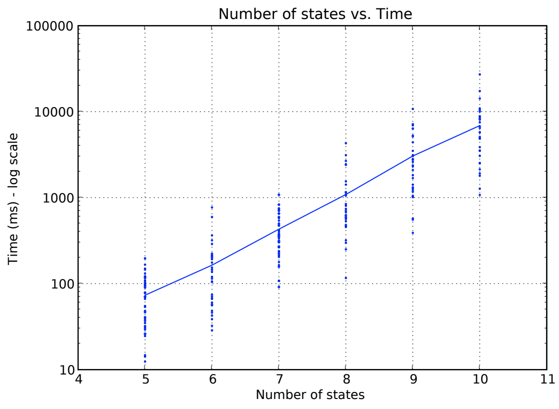

Figure 3: Scaling of ICG algorithm: number of states vs time

imax regret computation has very favorable anytime behaviour, as exhibited in Figure 2. During the iterative constraint generation procedure the regret gap shrinks extremely quickly at the beginning and slow progress is made near the end. If one does not need an exact measure of minimax regret, this anytime property may allow for fast approximations. This is especially true given that we want to use minimax regret to suggest queries (especially early on); robust decisions are made only at the end of the elicitation process. Detailed examination of anytime schemes is an important next step in this work.

5.2 Elicitation Effectiveness

We analyzed the effectiveness of our regret-based elicitation procedure by comparing it with the maximin criterion. We implemented a variation of the Double Oracle maximin algorithm developed by McMahan, Gordon & Blum (2003). The computation time for maximin is significantly less the that of minimax regret—this is expected since maximin requires only the solution of a pair of linear programs.

We used both maximin and minimax regret to assess each step of the preference elicitation procedure and paired each with the current solution and halve the largest gap query strategies. Thus we essentially have four algorithms: MMR-HLG, where policies are computed using regret and queries by HLG; MMR-CS, regret-based policies and queries using current solution; and MM-HLG and MM-CS, in which policies are computed using maximin and queries using HLG and current solution, respectively. We compare each of these strategies using three criteria, and show their performance after each elicitation query to see how effective the queries are. We measure the performance of each policy using: (a) its maximin value given the current (remaining) uncertainty in the reward function (b) its max regret value given the current (remaining) uncertainty in the reward function; (c) its true regret (i.e., loss relative to the optimal policy for the underlying true reward function **r**, where **r** is used to generate query responses). The minimax regret measure is the most critical since it provides the strongest guarantees; but we compare to maximin value as well, since maximin policies are optimizing against a very different robustness measure. True regret is not available in realistic settings; but this gives us an indication of how good the resulting policies actually are (as opposed to a worst-case bound).

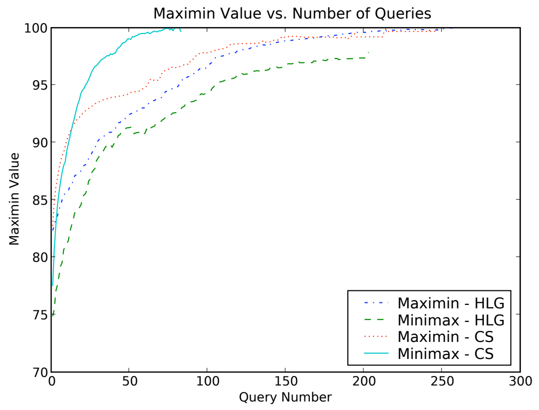

Figure 4: Preference Elicitation Comparison: Maximin Value

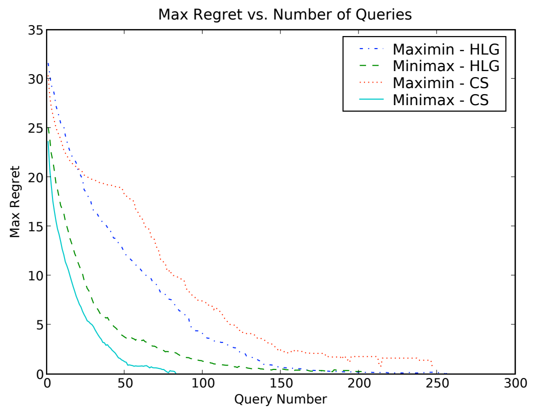

Figure 5: Preference Elicitation Comparison: Minimax Regret

Figures 4, 5 and 6 show the results of the comparison on each measure. The minimax regret criterion with the current solution selection heuristic performs extremely well on all measures. Somewhat surprisingly, it even outperforms the maximin criterion with respect to maximin value (except at the very early stages). Even though maximim is optimizing maximin value,

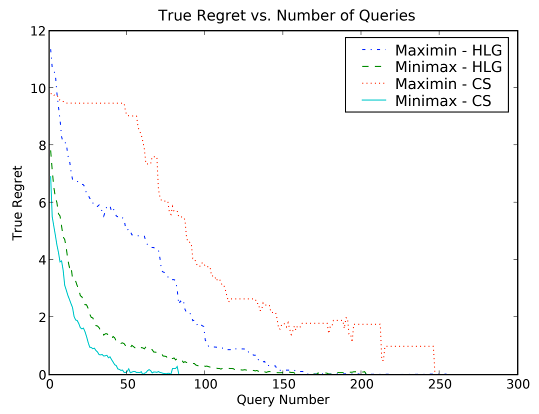

Figure 6: Preference Elicitation Comparison: True Regret

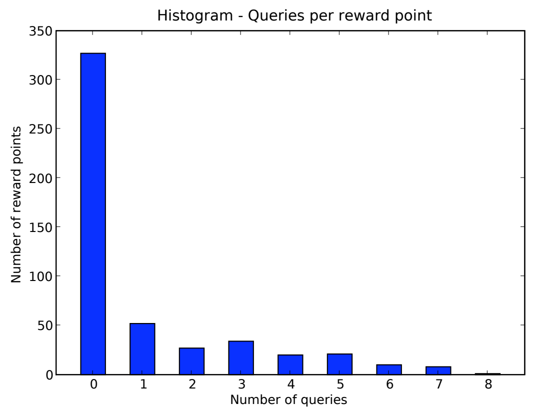

Figure 7: Histogram of the number of queries about each point in the reward space using MMR-CS.

the current solution strategy when paired with minimax policies is clearly asking much more informative queries, allowing for a larger reduction in the reward intervals at "highly relevant" state-action pairs. While the minimax policy does not optimize for maximin value, and hence cannot do as well before asking any queries, it provides better maximin value after only a few queries. This is clear when we examine how much reduction there is in the reward intervals over the course of the elicitation. Let χ measure the sum of the length of the reward intervals. At the end of the elicitation MMR-HLG has reduced χ to 15.6% of its original value (averaged over the 20 MDPs), while MMR-CS only reduces χ to 67.8 % of its original value. MMR-CS is effectively eliminating regret while leaving a large amount of uncertainty. Figure 7 illustrates this using a histogram of the number of queries asked by MMR-CS about each of the 1000 possible state-action pairs[6]. We can see that MMR-CS asks no queries about the majority of state-action pairs, and asks quite a few queries (up to eight) about a small number of "high impact" state-action pairs.

Figure 5 shows that MMR-CS is able to reduce regret to zero (i.e., find an optimal policy) after less than 100 queries on average. Recall that the MDP has 50 reward parameters (state-action pairs), so on average, less than two queries per parameter are required to find a provably optimal policy.

The minimax regret policies also outperform the maximin policies by a wide margin with respect to true regret (Figure 6). With the current solution heuristic, a near-optimal policy is found after fewer than 50 queries (less than on query per parameter), though to *prove* that the policy is near-optimal requires further queries (to reduce minimax regret).

It is worth noting that during the preference elicitation procedure, the HLG strategy does not require that the minimax regret actually be computed. Minimax regret is only necessary to assess when to stop the elicitation process (i.e., to determine in minimax regret has dropped to an acceptable level). the current level of regret). One possible modification to speed up the time required between queries, is to only compute the minimax regret after every k queries. Of course, using the HLG strategy will lead to a slower reduction in true regret and minimax regret as shown in Figures 5 and 6.

[6]1000 = 20 randomly generated MDPs with 10 states * 5 actions each.

6 Related Work

There has been a significant amount of work on robust MDPs which tackles the first step of our elicitation process, namely computing robust policies in the presence of uncertainty in the MDP model. Most of this work has centered around mitigating uncertainty over the transition function.

Iyengar (2005) shows how robust policies can be computed for uncertain transition functions using the maximin criterion by decomposing the problem across timesteps and using dynamic programming and an efficient suboptimization to find the worst case transition function. Bagnell, Ng & Schneider (2001) and Nilim & Ghaoui (2004) also use the maximin criterion with respect to different characterizations of uncertainty over transition functions with similar results. The topic of further eliciting information about transition functions is not addressed.

Delage and Mannor (2007) address the problem of uncertainty over reward functions (and transition functions) in the presence of prior information. They use the following percentile criterion which is less pessimistic than the maximin criterion for $\eta < 1$

$$\begin{aligned} &\underset{x,\pi}{\text{maximize}} \quad x \\ &\text{Subject to:} \quad \Pr\left[\mathrm{E}\left(\sum_{t=0}^{\infty}\gamma^t\tilde{r}(s_t)\,\middle|\, \pi\right) \geq x\right] \geq \eta \end{aligned}$$

For a given policy π, the percentile criterion results in an η guarantee that policy π will perform better than x^* , the optimal value of under the inuence of $\tilde{r}$ and $\tilde{P}$, where $\tilde{r}$ and $\tilde{P}$ are drawn from prior distributions. For values of $\eta = 1$ the above formulation reduces to the maximin criterion.

Delage and Mannor also contribute a method for eliciting rewards in their framework using monte carlo sampling to approximate the EVOI of gathering noisy information about a point in the space of rewards. The percentile approach is neither fully bayesian nor does it offer a real guarantee.

Zhang and Parkes (2008) present a setting that builds on inverse reinforcement learning which they call policy teaching. A *teacher* observes a *student* who is acting optimally with respect to an unknown reward function and provides an incentive which alter the behaviour of the student. The overall goal is to incent the student to execute the teacher's policy, however, the approach is essentially a form of reward elicitation which the queries are changes to the students incentives and the information gained is the change in the students behaviour.

7 Conclusions & Future Work

This work proposed an approach to specifying reward functions through regret-based elicitation. We showed how robust policies can be computed for Markov Decision Processes given partial reward information using the minimax regret criterion. We then showed how regret can be reduced by efficiently eliciting rewards information using bound queries. While the minimax regret criterion involves considerably more complex computation compared to similar criteria such as maximin, it offers an intuitive measure for driving elicitation and leads to a significant drop in true regret.

This work is preliminary and there are many directions in which to proceed. Perhaps the most interesting is the development of informative and intuitive queries that capture the sequential nature of the elicitation problem. Contingent policies can be conceptually taxing for a person to reason about, however, visitation frequencies offer a possible alternative way to refine reward with respect to the policy across time-steps. If the visitation frequencies that are rounded to integral values yield something like a count of the number of times a state and action are encountered. Queries could be constructed that ask a domain expert to trade-offs between *event counts* for significant parts of the reward space.

Another direction for improving quality of the elicitation is to incorporate *implicit* information in a manner similar to that of Zhang and Parkes (2008). The inverse reinforcement learning constraints developed by Ng and Russell (2000) can be used to translate observed behaviour into constraints on reward.

It will also be necessary to improve the efficiency of the minimax regret computation by exploiting structure using linear programming approaches to MDPs (Guestrin *et al.* 2003). Also, a tight upper bound on the minimax regret will still yield a guarantee and will likely offer a significant speedup.

Acknowledgements

The authors gratefully acknowledge the financial support of Canada's Natural Science and Engineering Research Council (NSERC).

References

Bagnell, J.; Ng, A.; and Schneider, J. 2001. Solving uncertain markov decision problems. *Tech Report.*

Benders, J. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik.*

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence* 170(8–9):686–713.

Boutilier, C.; Sandholm, T.; and Shields, R. 2004. Eliciting bid taker non-price preferences in (combinatorial) auctions. 204–211.

Boutilier, C. 2002. A POMDP formulation of preference elicitation problems. 239–246.

Chajewska, U.; Koller, D.; and Parr, R. 2000. Making rational decisions using adaptive utility elicitation. 363–369.

Delage, E., and Mannor, S. 2007. Percentile optimization in uncertain markov decision processes with application to efficient exploration. *ICML.*

French, S. 1986. Decision theory: an introduction to the mathematics of rationality. *Halsted Press.*

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research.*

Iyengar, G. 2005. Robust dynamic programming. *Mathematics of Operations Research.*

Keeney, R. L., and Raiffa, H. 1976. *Decisions with Multiple Objectives: Preferences and Value Trade-offs.* New York: Wiley.

McMahan, H.; Gordon, G.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. *ICML.*

Ng, A., and Russell, S. 2000. Algorithms for inverse reinforcement learning. *ICML.*

Nilim, A., and Ghaoui, L. E. 2004. Robustness in markov decision problems with uncertain transition matrices. *NIPS.*

Puterman, M. 1994. Markov decision processes: Discrete stochastic dynamic programming.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press.

Zhang, H., and Parkes, D. 2008. Value-based policy teaching with active indirect elicitation. *AAAI.*